(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105378721 B
(45)授权公告日 2019.02.19

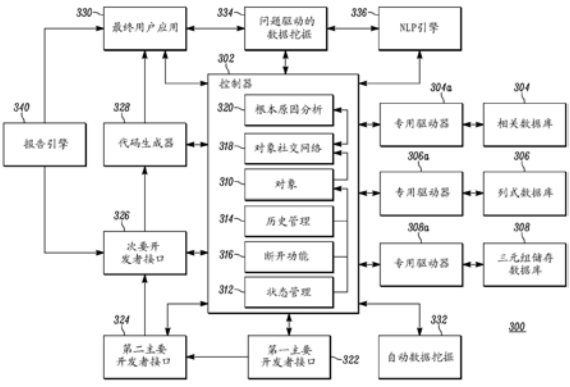
(21)申请号 201480027598.1
(22)申请日 2014.03.14
(65)同一申请的已公布的文献号
申请公布号 CN 105378721 A
(43)申请公布日 2016.03.02
(30)优先权数据
61/787177 2013.03.15 US
(85)PCT国际申请进入国家阶段日
2015.11.13
(86)PCT国际申请的申请数据
PCT/US2014/027854 2014.03.14
(87)PCT国际申请的公布数据
WO2014/143755 EN 2014.09.18
(73)专利权人 比乌拉工厂有限公司
地址 美国印第安纳州
(72)发明人 S.郭 R.E.约翰逊三世
B.R.希尔顿 D.A.内维尔
(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 王岳 刘春元
(51)Int.Cl.
G06F 16/242(2019.01)
G06F 16/25(2019.01)
G06F 16/22(2019.01)
G06F 16/28(2019.01)
(56)对比文件
CN 102880645 A,2013.01.16,
CN 102955697 A,2013.03.06,
CN 102855290 A,2013.01.02,
CN 102819600 A,2012.12.12,
US 2012317149 A1,2012.12.13,
US 2012317077 A1,2012.12.13,
US 2012303660 A1,2012.11.29,
US 2012271785 A1,2012.10.25,
审查员 王博实
权利要求书3页 说明书49页 附图4页

(54)发明名称

知识捕获和发现系统

(57)摘要

用于知识捕获和发现的系统包括知识库,其中所有输入数据被存储为对象和对象之间的关系,并且该输入数据可以根据多于一种存储格式进行存储。至少两个分级用户接口提供输入机构以获得输入数据、涉及输入数据的对象信息以及涉及输入数据的关系信息,由此准许开发最终用户应用。控制器从至少两个分级用户接口接收输入数据、对象信息和关系信息并且基于对象信息和关系信息使输入数据作为对象存储在知识库中。



CN 105378721 B

1.一种用于知识捕获和发现的系统,所述系统包括:

由至少一个处理设备实现的知识库,其中所有输入数据被存储为对象和对象之间的关系,并且其中输入数据可根据多于一种存储格式进行存储;

由至少一个处理设备实现的至少两个分级用户接口,其提供输入机构以获得输入数据、涉及输入数据的对象信息以及涉及输入数据的关系信息;以及

由至少一个处理设备实现并且操作连接到知识库和至少两个分级用户接口的控制器,其操作以从至少两个分级用户接口接收输入数据、对象信息和关系信息,并且基于对象信息和关系信息使输入数据作为对象存储在知识库中。

2.权利要求1的系统,其中知识库将所有输入数据存储在具有第一存储格式的第一数据库中,并且选择性地使输入数据的至少一部分从第一存储格式转换成至少一种第二存储格式并且存储在实现至少一种第二存储格式的至少一个第二数据库中,并且其中第一数据库包括三元组储存数据库。

3.权利要求2的系统,其中第二数据库包括如下中的任何一个或多个:相关数据库、列式数据库、图形数据库、关键值数据库、文档数据库和文件存储数据库。

4.权利要求2的系统,其中控制器基于输入数据的至少一个数据特性而指令知识库将输入数据的所述部分从第一存储格式转换成第二存储格式。

5.权利要求4的系统,其中输入数据的至少一个数据特性包括数据大小和数据模式中的任何一个或多个。

6.权利要求4的系统,其中输入数据的至少一个数据特性包括所要求的数据新鲜度和所要求的数据保持力中的任何一个或多个。

7.权利要求2的系统,其中控制器基于输入数据的至少一个使用特性而指令知识库将输入数据的所述部分从第一存储格式转换成第二存储格式。

8.权利要求7的系统,其中输入数据的至少一个使用特性包括如下中的任何一个或多个:数据写入的频率、数据更新的频率、数据读取的频率、数据读取请求类型和用户的并发性。

9.权利要求2的系统,其中控制器实现支持以非数据库依赖的格式表述的通用操作的统一接口,所述通用操作被转换成一种或多种数据库依赖的格式。

10.权利要求1的系统,其中至少两个分级用户接口包括至少一个主要开发者接口和次要开发者接口。

11.权利要求10的系统,其中至少一个主要开发者接口包括第一主要开发者接口,其包括支持软件组件的开发的集成开发环境。

12.权利要求10的系统,其中至少一个主要开发者接口包括第二主要开发者接口,其包括支持GUI应用和软件窗口小部件中的任何一个或多个的开发的基于图形用户接口(GUI)的平台。

13.权利要求10的系统,其中次要开发者接口包括操作以基于软件窗口小部件来构造应用元文件的图形用户接口。

14.权利要求13的系统,还包括:

操作连接到控制器和次要开发者接口的代码生成器,其操作以基于应用元文件来生成可执行应用。

15.权利要求1的系统，其中控制器还包括：

自然语言处理组件，其操作以基于人类可读的用户数据查询来生成机器可读的用户数据查询并且基于人类可读的应用规范来生成应用元数据。

16.权利要求1的系统，其中控制器还包括：

数据挖掘组件，其操作以基于机器可读的用户数据查询而从知识库检索所存储的数据。

17.一种用于经由至少一个用户接口表示计算机网络中的数据以促进对其的访问的方法，所述方法包括：

在由至少一个处理设备实现的控制器处，将经由由至少一个处理设备所实现的至少第一分级用户接口所接收的输入数据以第一存储格式存储在知识库的至少第一数据库中，输入数据被存储为对象和对象之间的关系；

通过控制器将存储在第一数据库中的输入数据的至少一部分转换成至少第二存储格式，并且将输入数据存储在知识库的第二数据库中，第一和第二存储格式是不同的存储格式；以及

通过控制器从第一分级用户接口或至少第二分级用户接口接收访问输入数据、涉及输入数据的对象信息以及涉及第一和第二数据库中的输入数据的信息的查询。

18.权利要求17的方法，还包括使用对象之间的相互关系以及使用网络发现技术来创建对象及其之间的关系的社交网络表示。

19.权利要求17的方法，其中转换步骤通过分析输入数据的更新和查询的模式而发起。

20.权利要求17的方法，还包括通过控制器更新第一和第二数据库中的输入数据。

21.权利要求17的方法，其中第一和第二数据库中的至少一个包括三元组储存数据库。

22.权利要求21的方法，其中第一和第二数据库中的至少一个包括如下中的任何一个或多个：相关数据库、列式数据库、图形数据库、关键值数据库、文档数据库和文件存储数据库。

23.权利要求17的方法，其中控制器基于输入数据的至少一个数据特性而将输入数据的所述部分从第一存储格式转换成第二存储格式。

24.权利要求23的方法，其中输入数据的至少一个数据特性包括数据大小和数据模式中的任何一个或多个。

25.权利要求23的方法，其中输入数据的至少一个数据特性包括所要求的数据新鲜度和所要求的数据保持力中的任何一个或多个。

26.权利要求17的方法，其中控制器基于输入数据的至少一个使用特性而将输入数据的所述部分从第一存储格式转换成第二存储格式。

27.权利要求26的方法，其中输入数据的至少一个使用特性包括如下中的任何一个或多个：数据写入的频率、数据更新的频率、数据读取的频率、数据读取请求类型和用户的并发性。

28.权利要求17的方法，还包括通过控制器实现支持以非数据库依赖的格式表述的通用操作的统一接口，所述通用操作被转换成一个或多个数据库依赖的格式。

29.权利要求17的方法，其中第一分级用户接口包括至少一个主要开发者接口，并且第二分级用户接口包括次要开发者接口。

30.权利要求17的方法，其中查询是人类可读的用户数据查询，并且所述方法还包括通过控制器基于人类可读的用户数据查询生成机器可读的用户数据查询以及基于人类可读的应用规范生成应用元数据。

31.权利要求17的方法，还包括通过控制器进行数据挖掘以基于机器可读的用户数据查询从知识库检索输入数据。

32.权利要求17的方法，还包括通过控制器监视查询和由其产生的数据模式。

33.一种用于经由至少一个用户接口表示计算机网络中的数据以促进对其的访问的方法，所述方法包括：

使用至少一个处理设备将输入数据存储在知识库中，输入数据作为对象和对象之间的关系被存储在知识库中，知识库包括至少两个数据库，每一个以不同存储格式存储输入数据；

通过由至少一个处理设备实现的至少两个分级用户接口提供输入机构以获得输入数据、涉及输入数据的对象信息和涉及输入数据的信息；以及

通过由至少一个处理设备实现的控制器从至少两个分级用户接口接收所述输入数据、对象信息和信息，以及通过控制器基于所述对象信息和信息使输入数据作为对象存储在知识库中。

知识捕获和发现系统

[0001] 相关申请的交叉引用

[0002] 本申请要求在2013年3月15日提交并且标题为“Enterprise Level Application Software Development System”的美国临时专利申请序列号61/787,177的优先权，该专利申请的教导通过该引用并入本文中。

技术领域

[0003] 本公开大体涉及企业信息管理，并且具体地涉及作为企业信息管理的一部分或针对企业信息管理的补充的、用于知识的捕获和发现的系统。

背景技术

[0004] 诸如此处共同地称为企业的商业或其它组织之类的各种类型的实体通常被创建、组织和操作以便实现特定目标，例如向相关消费者提供物品和/或服务。为了实现这些目标，不同规模的许多企业共享参与众多过程中以及在执行这样的过程时获得与此相关的大量数据的特性。随着企业变得越来越大和/或设法实现前所未有地更困难和复杂的目标，得到为了恰当地管理这样的过程所牵涉的过程的真实理解以及实现它们所要求的资源的能力常常变为棘手的问题。尽管可能存在可以用于开发这样的洞察(insight)的大量数据，但是这样的数据的绝对的数量、复杂性和可变性使得难以利用该潜在资源。

[0005] 当前存在各种技术以解决该问题的部分。例如，为了高效地存储并提供对数据的访问，在过去四十年内已经开发了众多数据库技术，其中每一种可以具有特定的优点和缺点。附加地，甚至在这样的技术的情况下，为决策做出者提供对该数据的访问要求专门受训的技术人员的支持，诸如软件开发和/或数据库管理专家。这导致大开销以及未满足数据消费者的需要的非常真实的可能性。又进一步地，即便技术对于来自这样的存储数据的递送报告是已知的，开发关于这样的数据所表示的过程的理解和洞察的能力也保持为困难的任务。

[0006] 因此，将有利的是，提供一种准许捕获企业数据并且在此之后以促进对其访问的方式使得可用的系统，使得甚至具有很少企业数据管理经验或没有这种经验的人员也可以能够开发之前实现起来过分昂贵(如果不是不可能的话)的洞察。

发明内容

[0007] 本公开描述一种克服现有技术解决方案的缺点的用于知识捕获和发现的系统。具体地，该系统包括知识库，其中所有输入数据被存储为对象和对象之间的关系。附加地，输入数据可以根据多于一种存储格式进行存储。该系统中的至少两个分级用户接口提供输入机构以获得输入数据、涉及输入数据的对象信息和涉及输入数据的关系信息，由此准许开发最终用户应用。附加地，操作连接到知识库和至少两个分级用户接口的控制器从至少两个分级用户接口接收输入数据、对象信息和关系信息，并且基于对象信息和关系信息而使输入数据作为对象存储在知识库中。

附图说明

[0008] 在随附权利要求中具体地阐述本公开中所描述的特征。这些特征将从结合附图进行的以下详细描述的考虑变得显而易见。现在仅通过示例的方式参照附图描述一个或多个实施例，其中相似附图标记表示相似元件，并且其中：

[0009] 图1是可以用于实现本公开的各方面的示例性处理设备的框图；

[0010] 图2是图示了可以用于实现本公开的特征的各联网硬件组件的框图；

[0011] 图3是图示了依照本公开的各实施例的功能组件的框图；并且

[0012] 图4是基于RDF和相关数据的数据转换处理的示例性实现的框图。

具体实施方式

[0013] 图1图示了可以用于实现本公开的教导的代表性处理设备100。处理设备100可以用于实现例如以下更详细描述的系统200的一个或多个组件。例如，处理设备100可以包括工作站计算机或服务器计算机。不管怎样，设备100包括耦合到存储组件104的处理器102。存储组件104继而包括存储的可执行指令116和数据118。在实施例中，处理器102可以包括能够执行存储的指令116并且对存储的数据118操作的微处理器、微控制器、数字信号处理器、协处理器等中的一个或多个或者其组合。同样地，存储组件104可以包括一个或多个设备，诸如易失性或非易失性存储器，包括但不限于随机存取存储器(RAM)、只读存储器(ROM)或其它非暂时性机器可读设备。又进一步地，存储组件104可以以各种形式来体现，诸如硬盘驱动器、光盘驱动器、软盘驱动器等。图1中所图示的类型的处理器和存储布置对本领域普通技术人员是公知的。在一个实施例中，本文描述的处理技术被实现为一个或多个处理设备100的存储组件104内的可执行指令和数据的组合。

[0014] 如所示出的，设备100可以包括与处理器102通信的一个或多个用户输入设备106、显示器108、外围接口110、其它输出设备112和网络接口114。尽管处理器102和各其它设备/显示器/接口106-114之间的连接被图示为分离的直接连接，但是本领域普通技术人员将认识到，在实践中，一个或多个总线子系统(未示出)可以用作用于如所意图的那样使处理设备100的各组件彼此通信的机构。用户输入设备106可以包括用于向处理器102提供用户输入的任何机构。例如，用户输入设备106可以包括键盘、鼠标、触摸屏、麦克风和适合的语音识别应用或任何其它构件，由此设备100的用户可以向处理器102提供输入数据。显示器108可以包括任何常规显示机构，诸如阴极射线管(CRT)、平坦面板显示器或本领域普通技术人员已知的任何其它显示机构。在实施例中，显示器108与由处理器102所执行的适合的存储指令116结合可以用于实现如下文所描述的图形用户接口。以该方式的图形用户接口的实现对于本领域普通技术人员是公知的。外围接口110可以包括用于与各种外围设备通信所必要的硬件、固件和/或软件，所述各种外围设备诸如媒体驱动器(例如磁盘或光盘驱动器)、其它处理设备或与本技术结合使用的任何其它输入源。同样地，(多个)其它输出设备112可以可选地包括能够向设备100的用户提供信息的类似媒体驱动器机构、其它处理设备或其它输出目的地，诸如扬声器、LED、打印机、传真机、触觉输出等。最后，网络接口114可以包括允许处理器102经由如本领域中所已知的有线或无线网络(不管是局域的还是广域的，专用的还是公用的)与其它设备通信的硬件、固件和/或软件。例如，这样的网络可以包括万

维网或互联网、或者专用企业网络，如本领域中所已知的。

[0015] 尽管设备100已经被描述为用于实现本文描述的技术的一种形式，但是本领域普通技术人员将认识到，可以采用其它功能等同的技术。例如，如本领域中所已知的，经由由一个或多个处理器所执行的可执行指令而实现的一些或全部功能也可以使用诸如专用集成电路(ASIC)、可编程门阵列、状态机等之类的固件和/或硬件设备来实现。此外，设备100的其它实现可以包括比所图示的那些更多或更少数量的组件。再次地，本领域普通技术人员将认识到可以使用的为数众多的变化是该方式。又进一步地，尽管在图1中图示了单个处理设备100，但是要理解到，这样的处理设备的组合可以被配置成结合地操作(例如使用已知联网技术)以实现本公开的教导。由于处理设备和网络的不断改变性质，图1中所描绘的处理设备100的描述仅意图为代表本领域普通技术人员所已知的大量的处理设备的特定示例。

[0016] 现在参照图2，示出系统200，其图示了可以用于实现本公开的教导的许多硬件组件。如所示出的，系统200包括控制器202，其可以包括一个或多个服务器计算机。控制器202直接地或者经由一个或多个网络204与各种其它组件通信。网络204可以包括如本领域中所已知的无线或有线网络的任何期望的组合，不管是局域的还是广域的，专用的还是公用的。如上文指出的，这样的网络可以包括万维网或互联网或专用企业网络，如本领域中所已知的。

[0017] 可以包括诸如桌上型或膝上型计算机或移动计算设备之类的处理设备的工作站206可以经由网络204与控制器202通信。在实施例中，工作站206可以实现能够提供图形用户接口的web(网络)浏览器应用或其它应用，如本领域中所已知的。使用这样的应用，工作站206还可以实现若干分级用户接口中的一个，如下文更详细描述的。附加地，工作站206可以操作成接收和执行基于这样的分级用户接口所开发的一个或多个最终用户应用。

[0018] 如进一步示出的，一个或多个分级用户接口服务器208可以与控制器202通信，并且经由网络204与工作站206通信。如本领域中所已知的，一个或多个分级用户接口服务器208可以包括应用和web服务器的组合，其中web服务器服务于来自用户的请求以使用与web服务器通信的应用服务器所提供的应用资源来执行动作。具体地，web服务器将这样的请求中继给应用服务器，其采取指定动作并且将该动作的结果返回给web服务器，其继而将结果中继给用户工作站206。应当指出的是，尽管这样的web服务器可以被视为硬件组件(如本文所描述的任何服务器那样)，但是这样的web服务器也可以是在计算机系统上操作的软件模块。

[0019] 不管怎样，依照这样的技术，分级用户接口服务器208可以提供至少一个主要开发者接口和/或次要开发者接口，如下文更详细描述的。例如，分级用户接口服务器208可以实现web页面等，其显示在工作站206上以实现一个或多个分级用户接口。这些分级接口继而可以在一个实施例中用于最终开发应用元文件。如本文中所使用的，应用元文件可以包括足以生成可执行源代码的信息，诸如如本领域所已知且在下文描述的用户接口标记(markup)或功能标记。(多个)最终用户应用服务器212可以包括web和应用服务器(如上文描述的)，将由代码生成服务器210所生成的最终用户应用提供给请求用户的功能。

[0020] 如图2中进一步示出的，控制器202与共同地建立数据库联合体(complex)219的多个数据库服务器214-218通信。如本文所使用的，数据库可以包括实现已知数据库存储格式

的任何适合存储设备，包括但不限于本文中指出的各种数据库存储格式。例如，可以提供实现第一存储格式或模式的一个或多个第一数据库服务器214、实现第二存储格式或模式的一个或多个第二数据库服务器216、以及这样往上直到实现第N存储格式或模式的一个或多个第N数据库服务器218。例如，在一个实施例中，第一数据库服务器214可以实现所谓的三元组储存(triplestore)数据库，而第二数据库服务器216可以实现相关数据库，并且第N数据库服务器218可以实现又另一数据库存储格式，诸如但不限于列式数据库、图形数据库、关键值数据库、文档数据库和文件存储数据库。如本领域普通技术人员将领会到的，可以使用又其它数据库存储格式，并且本公开不在这方面受限。

[0021] 以此方式配置，每一种数据库存储格式的相对优点是可用的，并且如下文更详细描述的控制器202有效地充当抽象层以保护最终用户免于不必精通每一种数据库存储格式的复杂性。在下文同样描述的一个实施例中，控制器202操作成如所需要地发起从一种存储格式向另一种的数据的转换以改进总体性能。在另一实施例中，多种数据库存储格式的存在准许用户具体地限定条件，从而引起数据的转换。例如，在所谓的CAP(一致性、可用性、分区容忍性)定理之下，断言到，利用分布式数据库，人们可能仅具有三个属性之中的两个：一致性(所有节点具有最新且相同的信息)、可用性(正常运行时间/采取请求)和分区容忍性(处理断开状态)。基于该目标，用户可以指定针对各种数据库之间的数据转换的要求以使这些属性中的每一个或其任何组合优化。

[0022] 如进一步示出的，控制器202可以经由网络204与一个或多个自然语言处理(NLP)服务器220和一个或多个数据挖掘服务器222通信。如下文更详细描述的，NLP服务器220操作成不仅在访问数据库联合体219内的数据时而且在开发最终用户应用时促进自然语言查询的使用。与NLP服务器220结合地工作，数据挖掘服务器222基于存储在数据库联合体219中的数据而实现各种数据挖掘任务，诸如根本原因分析、分类、集群、相关联规则发现和/或回归分析。

[0023] 现在参照图3，图示了根据本文提供的各种功能的系统300。要指出的是，图3中所图示的每一个组件可以使用如上文描述的一个或多个处理设备来实现，从而实现本文描述的功能。在系统300内，控制器302与在所图示的示例中包括相关数据库304、列式数据库306和三元组储存数据库308的多个数据库304-308通信。如本领域中所已知的，每一个数据库304-308可以(并且通常将)包括其自身的促进与数据库的交互的数据库管理系统(DBMS)。如所示出的，控制器302通过由对应DBMS实现的应用编程接口(API)304a-308a与各数据库304-308通信。这样的API可以由制造商专有驱动器或专有表述性状态转移(REST)接口体现。

[0024] 在实施例中，由系统200、300处理的每一个数据片段被视为对象。因而，每一个数据片段被提供有唯一地标识对象的对象标识、阐述对象的当前状态的状态指示符、指示相对于针对对象的修订的序列的当前修订状态的修订号码、以及指示特定修订何时创建的时间戳。对象在系统中永远不被物理地删除。当由用户修改或“删除”对象时，系统简单地创建对象的修订以反映其当前状态。旧修订保持为历史记录。在以下的表1中使用公知的Java脚本对象符号(JSON)格式示出对象的示例，在该情况下是可以在图形用户接口中找到的类型的提交按钮，其中根据许多名称-值(name-value)对来描述对象：

```
{
    "id": "jk234hjk34h2i3o4u89ghkjnhk",
    "objectType": "widget",
    "widgetType": "button",
    "title": "submit",
    "history": {
        "rev": "12",
        "state": "active",
        "timestamp":"1394654029"
    },
    "widgetProperties": {
        "width": "20px",
        "height": "15px",
        "x": "100px",
        "y": "150px",
        "float": "left"
    },
    "behavior": [
        {
            "event": "single click",
            "action": "asdfjk314j2hjwdflj234"
        }
    ]
}
```

[0025] [0026]

[0027] 表1。

[0028] 在该示例中，对象属于“窗口小部件(widget)”类型，并且进一步地是“按钮”类型的名为“提交”的窗口小部件。该对象当前是“活动”的并且在其第十二次修订上。其还包括行为定义，特别地在“单击”事件中采取什么“行动”。如本领域中已知的，JSON表示不仅对于人类是可理解的，而且还可以由机器解析。如本领域技术人员将领会的，可以使用各种各样的对象类型和子类型来将任何数据片段事实上视为对象。例如，提供给系统200、300的自然语言查询可以被视为一系列“文字”对象，其中查询本身被视为包括这样的“文字”对象的集合的对象。在另一示例中，软件源代码的区段可以被视为包括许多“声明”、“运算符”、“变量”、“变量名”等对象的第一对象。

[0029] 将系统中的所有数据视为对象的优点在于，其与“三元组”数据表示概念相兼容，其中可以在本公开的上下文中做出关于对象之间的关系的声明。例如，所谓的资源数据框

架(RDF)规范建立主语-谓语-宾语表述(三元组)以便做出涉及“资源”(例如web资源)的声明,尽管该概念容易地适用于本文所使用的意义下的对象。作为简单示例,构建于以上所指出的示例上,以web形式使用的按钮窗口小部件的事实可以根据以下表2中所图示的三元组来描述:

[0030] x:button y:is_in z:form c:91fbc220-aacd-11e3-a5e2-0800200c9a66

[0031] 表2。

[0032] 在该示例中,主语(button)通过关系谓语“is_in”与宾语(form)相关。如本领域中已知的,在RDF中,前缀x、y和z通常是提供唯一地命名实体的信息的统一资源标识符(URI)的简写表示,在该示例中是“button”、“is_in”和“form”。在目前优选实施例中,该三元组形式被扩展成所谓的“nquad”格式,其提供针对上下文的附加字段(具有前缀c)。因而,在表2的示例中,该上下文字段用于具有将对象数据链接在一起的通用唯一标识符(UUID)值。也就是说,在该实施例中,上下文四元组(quad)字段在单个对象中将各种数据绑在一起,其在实际中可以包含数千个三元组/四元组值。不管怎样,如下文更详细描述的,比如RDF的协定还提供传达本体论(ontology)信息的声明,即描述用于组织信息由此提供知识表示的结构框架的信息,该本体论信息可以用于协助从一种存储格式向另一种的数据的转换。

[0033] 在实施例中,所有数据经由控制器302被添加至数据库304-308、在其中改变、从其中读取或从其中删除,如上文指出的,所述控制器302终止所有数据库特定协议以使得控制器302的用户仅被呈现以单个接口。具体地,单个接口可以支持以不依赖于任何一种数据库存储格式的格式表述的通用(common)操作。例如,控制器302可以为最终用户提供统一API以使用基于JSON的结构化查询语言(SQL)类API来管理数据。SQL类API促进与系统300的外部和内部用户二者的通信,特别地因为其将严谨且严格的相关数据库要求桥接到相对松散且灵活的NoSQL数据库要求,由此使得传统开发者能够享用NoSQL数据库或多个数据库的益处而不经受陡峭学习曲线。为了完整性,可能合期望的是,在某些实例中为最终用户(除SQL类统一API之外)提供对每一个数据库304-308的DBMS的访问,尽管预期到对底层数据库API的这样的访问将对于缺乏这样的API的具体知识的最终用户不是优选的。不管怎样,在该实施例中,SQL类统一API方法包括通常由所有数据库管理系统提供的创建、读取、更新和删除(CRUD)操作。这样的创建、读取、更新和删除操作的JSON示例在以下表3-6中图示。

[0034]

```
{
    “collection”:”VideoRental”,
    “data”:{
        {“name”:”Customer”, “CustomerFirstName”:”Paul”, “CustomerId”: “9001”,},
        {“name”:”Rented”, “RentalDate”:”09/28/01”},
        {“name”:”Video”, “VideoId”:”14564”}
    }
}
```

[0035] 表3. - JSON创建

```
{
    "collection":"VideoRental",
    "select": "CustomerFirstName",
    "where":{
        "relation": {"name": "Rented"},
        "object": {"VideoId":"14564"}
    }
}
```

[0036]

表4. - JSON读取

```
{
    "collection":"VideoRental",
    "update": "CustomerFirstName",
    "where":{
        "relation": {"name": "Rented"},
        "object": {"VideoId":"14564"}
    }
    "value":"Jane"
}
```

[0038]

[0039]

[0040] 表5. - JSON更新

```
{
    "collection":"VideoRental",
    "where":{
        "relation": {"name": "Rented"},
        "object": {"VideoId":"14564"}
    }
}
```

[0041]

[0042] 表6. - JSON删除。

[0043] 本领域技术人员将认识到，表3-6中的图示是SQL类统一API的示例，并且进一步地，相同的SQL类统一API可以以诸如XML之类的其它格式实现。基于这样的操作请求，在以上示例中，控制器302将JSON请求转换成必要的数据库特定的查询格式。例如，构建于以上所图示的操作上，用户可以将如表4中的读取请求提交至控制器302。在查询三元组储存数据库308时，控制器302将形成以下表7中所图示的类型的SPARQL查询：

[0044]
```
SELECT ?x
FROM VideoRental
WHERE
{ ?x ?y ?z
    WHERE
    {
       ?y name Rented.
       ?z has property ?h
           WHERE
           {
              ?h name VideoId.
              ?h value 14564.
           }
    }
}
```

[0046] 表7。

[0047] 在该示例中，映射规则是："collection":"X" => FROM X; "select":"X" => SELECT ?x; "relation":{...} => WHERE {?x ?y ?z WHERE {?y ...}};等等。该类型的进一步映射将可由本领域普通技术人员容易导出。

[0048] 当添加数据(涉及对象，如上文描述的)时，控制器302首先使数据以如上文描述的三元组的形式来添加，即其最初在三元组储存数据库308中首先创建并且针对这样的数据的查询至少最初被应用于三元组储存数据库308。在实施例中，三元组储存数据库308可以坚持其中第四个元素被添加到三元组的所谓的nquad格式；在该情况下，第四个元素是如上文描述的对象化标识符。

[0049] 当用户查询数据时，在控制器302中实现的查询解析器或监视器监视查询和结果所得的数据模式。这样的查询解析器是本领域中已知的，如例如在Applications Manager by Zoho Corporation Pvt. Ltd.(在http://www.manageengine.com/products/applications_manager/database-query-monitoring.html处可获得)中提供。例如，可以针对具体关键性能指示符监视所有查询，包括但不限于什么对象在被访问、是在将数据写入到其还是从中读取、所讨论的数据大小、查询的频率(如从存入数据所外推的)或在执行什么具体类型的报告/SELECT声明(同样地，如从存入数据所外推的)。作为结果，查询解析器能够将现有查询模式匹配到预定义的数据变换触发规则，其示例在下文提供。这些规则被设计成使得当数据模式满足给定规则的条件时，将数据从一种存储格式变换成另一种(部分地或整体地)的需要被检测。也就是说，预定义的变换规则准许控制器302决定是否可以变换某些数据；如果其可以被变换，则控制器302发起变换过程，其循环访问原始数据(即以第一数据存储格式存储的)并且以目标或第二数据存储格式创建新数据。同时，原始数据

保持不被触碰使得用户仍然可以在变换过程期间针对数据进行查询。一旦数据被变换，则将变换过程告知给查询解析器以使得查询解析器可以改变其针对该数据部分解析未来的查询的方式。例如，在实施例中，查询解析器修改其将SQL类统一API操作映射到特定底层数据库API的方式，使得未来的查询将被正确地处理并且将返回正确的回答。

[0050] 可能存在其中不知道哪种数据库存储格式将对于给定数据部分是最佳的情况。在这些实例中，可能合期望的是，将对象变换成每一种可用的数据库存储格式并且执行模拟负载测试。这样的负载测试可以基于所收集的日志数据而模仿真实世界的用户动作。当进行这样的负载测试时，监视各种活动的性能，并且可以根据各种数据库存储格式中的哪一种展现出如由任何合期望的准则所评定的最佳性能来选择“最佳”数据库存储格式。如果例如结果指示明显的性能改进，则可以创建附加规则以使得其通过涉及相关类型数据的数据查询而触发。在可替换实施例中，可以采用已知机器学习技术来推理这样的新规则。例如，机器学习算法可以使用已知规则来训练统计模型，其继而可以用于推理新的之前未知的规则。以此方式，可以避免针对另外未知的数据的性能测试(其可能是耗时的过程)，并且作为代替基于立即推理的规则而直接变换。在此之后，如果期望并且假定可用资源，则所推理的规则可以通过更准确的模拟负载测试来进一步验证。

[0051] 如上文指出的，可以采用规则来确定控制器302何时应当发起数据变换。在实施例中，可以考虑各种因素以建立这样的规则，所述因素可以一般地分组成数据因素或特性和使用因素或特性。数据特性涉及可能影响最优数据库存储格式的确定的底层数据的具体属性，并且包括但不限于数据大小、所要求的数据新鲜度或所要求的数据保持力。使用特性涉及如何使用数据的属性，并且可以包括但不限于数据写入的频率、数据更新的频率、数据读取的频率、数据读取请求类型和用户的并发性(concurrency)。

[0052] 关于各种数据特性，数据可以是相对短的以几字节测量的简单文本值、以兆字节测量的图形、或者大小为十亿字节的视频。如本领域中已知的，每一个图形的大小可以确定哪种类型的数据库将最佳地适合于其存储。另一相关数据特性是数据的所要求的“新鲜度”。例如，如本领域中已知的，数据库304-308的每一个可以实现某种形式的数据高速缓存。报告数据的临时高速缓存虑及大数据改进，但是其仅在报告内的数据不如数据被访问那样经常改变时是可行选项。又另一相关数据特性是所要求的数据保持力。在该情况下，数据通常仅直接用于某一时间段。例如，逐秒生产线数据通常将在未来的几周或几月内不是直接有用的。因此，可能合期望的是做出优化选择，其中在给定相对低的使用频率的情况下，将数据从昂贵但快速的数据库存储机构自动存档到较慢但低成本的存储机构。

[0053] 关于各种使用特性，可以采用数据读取、写入和/或更新的频率。例如，某些数据依赖于其类型可以一年写入一次(诸如年度报告的创建中所涉及的数据)，或者其可以在生产线的情况下是每秒许多次。相关地，一些数据一旦被写入就将永不改变，而其它数据可以频繁地改变。如果低频率数据被复制在多个区域中，则其更新将花费逐渐更长时间以沿着线成链。此外，许多系统具有数据读取对数据写入之间的权衡，即一个操作比另一个更为消耗资源的。又进一步地，如本领域中已知的，即便在高频率的数据读取的情况下，其产生较大的差异，如果给定报告使用相同集合的索引准则的话。例如如果你在看针对竞争比赛的高分列表，则可以每秒对其进行读取。然而，从比赛高分到具体划分高分的改变可能永不改变，或者极不频繁地改变。进一步关于报告场景，用户的并发性可能对确定最佳存储格式具

有显著影响。例如，如果存在一个用户运行报告，则对报告进行高速缓存因此其驻留在存储器中将不会提供明显的性能改进。然而，如果100个人每秒请求相同报告，则底层数据的高速缓存将引起明显的性能改进。

[0054] 可以基于这些特性开发各种规则。基于数据的性能可以通过在数据库之间转换或者在相同数据库中管理数据而改进。例如，如果存在高频率写入(更新)数据，则可能有利的是使用所谓的大数据宽列式数据库。为此目的，可以监视针对基于列的数据的查询。如果查询对未索引的列重复地运行，则可能需要创建次级索引。可替换地，如果在某一时间段之后查询不再使用具体索引，则可以移除索引。

[0055] 在另一示例中，如果底层数据模型是基于关键值对的集合，则应当使用文档存储引擎。因此，可以创建规则以例如寻找看起来是阵列内的阵列的数据结构。相关地，某些二进制数据(诸如照片或视频)将最佳地存储在基于文件的数据库系统中。关于关键值储存使用场景，控制器302虑及暴露还链接到存储在分离接口中的相关数据的本机二进制数据接口。例如，可以存在用于视频的对象类型。如以上对象示例中那样，每一个这样的视频具有唯一关键标识符，其链接到存储在基于文件的数据库中的二进制对象文件，但是其它元数据存储在相关数据库中。

[0056] 如果数据要求高度坚持所谓的ACID(原子数、一致性、隔离、持久性)性质，则具有约束的相关数据库将是最佳适合的。然而，甚至在该场景中，应当分析某些权衡以确定最佳拟合。例如，由于交易的高并发性和绝对大量，所以来自银行自动取款机(ATM)的数据是基于BASE(基本上可用、软状态、最终一致性)模型而不是ACID，其可以使用宽列式数据库更好地实现。

[0057] 对于其中底层数据模型描述任何类型的网络、图形、对象之间的连接等的数据，然后这样的数据将最佳地存储在图形数据库中。在该情况下，可以建立规则以搜索暗示许多关系(例如外来关键关系)的查询模式，如本领域中已知的，其涉及时间上非常昂贵的相关数据库中的多个结合操作。

[0058] 在又另一示例中，如果存在例如给定报告查询的高度重复，则将有益的是使用高速缓存(不管底层数据库存储格式如何)。如本领域中已知的，高速缓存规则确定高速缓存中的数据有多频繁地改变并且高速缓存失效可以是基于时间的和/或在针对源数据发生改变时具有失效能力。在该实例中，高速缓存的数据可以存储为其自身的分离的对象。例如，高速缓存对象的源数据可以以宽列式数据库存储格式驻留，但是在转换之后，实际高速缓存的数据可以以关键值存储格式存储在高速缓存存储器中。

[0059] 如上文描述的，所有数据最初存储在三元组储存数据库308中，并且控制器302确定何时要求从三元组储存格式到另一格式的转换，或者反之亦然。在实施例中，将数据从第一数据库存储格式转换到第二数据库存储格式的实际过程可以就所有数据至少最初以三元组储存数据库格式存储的方面包括在三元组储存数据库308的功能内。因而，将要求从另一数据库存储格式到三元组储存数据库存储格式以及从三元组储存数据库存储格式到另一数据库存储格式的格式转换二者。必要地，用于给定转换的特定技术将依赖于第一数据库存储格式和目标或第二数据库存储格式或源的性质。

[0060] 一般地，到三元组储存数据库存储格式的转换是基于标识以源数据库存储格式的最基本或基础的数据结构并且将那些数据结构映射到三元组。例如，当从关键值存储格式

向三元组储存存储格式转换时，转换过程(诸如所详述的RDF，如以下另外的示例中所描述的)可以循环访问每一个关键值并且做出对应三元组。当从宽列式存储格式向三元组储存存储格式转换时，转换过程可以循环访问每一个关键空间、列族、列和行，从而沿途形成三元组。当从文档存储格式向三元组储存存储格式转换时，转换过程可以循环访问每一个类集、文档和关键值，从而沿途形成三元组。当从图形数据库存储格式转换时，转换过程可以通过遵循其之间的连接并且沿途形成三元组而循环访问数据中的所有节点。当从相关数据库存储格式转换时，转换过程最初循环访问每一个表格，并且针对每一个表格建立其中谓语固定为“is a table of”的三元组。同样地，在每一个表格中标识任何外来关键关系或其它索引或性质，并且以三元组的形式包括它们，例如“x:table1.column1 y:is_foreign_key_to z:table2.column2.”。在每一个表格内，转换过程还循环访问每一个列。每一个列基于固定的三元组谓语"is a column of"而以三元组格式首先定义，其中三元组主语为列名称并且三元组宾语为包含在给定单元内的实际数据值。同样地，转换过程循环访问每一行，其中行内的每一个单元变为其自身的三元组。

[0061] 以类似方式，从三元组储存数据库存储格式向另一数据库存储格式的转换基本上基于三元组。如上文指出的，在三元组储存数据库存储格式是以nquad形式并且因此包括含有对象标识的第四个元素的情况下，对象标识用于建立要转换的三元组数据的上下文。因而，当从三元组储存存储格式向关键值存储格式转换时，每一个三元组被转换为关键值。当从三元组储存存储格式向宽列式存储格式转换时，转换过程首先标识三元组数据中的所有不同的谓语并且创建用于每一个的列族。在此之后，转换过程循环访问每一个三元组并且形成用于每一个的行。基于先前的查询信息(如例如由控制器302中的查询解析器所提供的)，用于所转换的数据的索引模式可以基于其先前的使用而导出。用于导出这样的索引模式的技术在本领域中是已知的，如例如在“Oracle Database Performance Tuning Guide (11g Release 1(11.1): Automatic SQL Tuning” (在http://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm#PFGRF028可获得)中所教导的。在此之后，如所需要的次级索引可以基于所导出的索引模式来创建。当从三元组储存存储格式向文档存储格式转换时，首先分析所转换的三元组数据中的所有三元组以标识对应于文档的谓语(例如“is_contained_in”)。在此之后，转换过程循环访问每一个三元组并且基于每一个三元组创建关键值条目，该关键值条目然后链接到对应文档中。当从三元组储存存储格式向图形存储格式转换时，转换过程可以循环访问三元组并且构建出顶点和边。

[0062] 除以上描述的控制器发起的转换之外，要认识到大量数据存储在已经存在的RDF数据库中。为了使用这些现有数据库，在三元组储存数据库308中提供将这样的预先存在的RDF数据转换成相关数据的能力。出于该描述的目的，假定三元组数据坚持RDF格式，尽管也可以使用其它三元组格式。具体地，外部RDF数据的转换开始于创建表格，该表格具有两个默认列：标识列，其充当用于表格的主要关键，包括从1开始的连续整数；以及资源名称列，其包括指定资源的名称(因为该术语通常在RDF用法中使用)的字符串。从该基本表格，标识三元组数据内的几乎所有性质(谓语)并且将其转换成表格内的列。不是所有RDF性质都以该方式使用，因为一些性质(本文称为元性质)提供关于数据的底层本体论结构的信息，而不是其本身的语义数据，该本体论信息可以用于进一步开发所转换的三元组数据的相关数据库表示。使用RDF性质扩展表格可以通过使用简单示例来进一步解释。

[0063] 以下表7阐述许多RDF声明：

[0064]
<lord of the rings> <subject> <middle earth story>.

<lord of the rings> <author> <J. R. R. Tolkien>.

<lord of the rings> <pages> <4709>.

<a song of ice and fire> <subject><seven kingdoms>.

<a song of ice and fire> <author><George R.R. Martin>.

<a song of ice and fire> <pages><4674>.

[0065] 表7。

[0066] 按照以上指出的涉及使用性质来标识附加表列的转换原理，表7中的RDF声明可以被转换为以下表8中所示的关系表示。

[0067]

id	资源名称	主语	作者	页
1	lord of the rings	middle earth story	J.R.R.Tolkien	4709
2	a song of ice and fire	seven kingdoms	George R.R. Martin	4674

[0068] 表8。

[0069] 如该示例展示，RDF向相关数据的转换是数据结构或元数据而不是数据本身的转换。为了进一步开发转换过程，将有利的是利用RDF元性质中所找到的元性质。

[0070] RDF和关系存储格式共享类似数据视图，因为它们均依赖于类和实例视图。另一方面，在RDF中，类和实例由所保留的元性质清楚地定义和支持，诸如rdf:class, rdf:type, rdfs:domain, rdfs:range等。另一方面，在关系格式中，尽管类/实例视图未明确地定义，但是其以称为“表格和元组”的另一形式有效地实现。表格可以被视为类，而列可以被视为类性质并且元组(行/记录)被视为实例。因而在实施例中，将RDF格式化数据转换成关系格式化数据的方案依赖于将RDF类转换成关系表格并且将RDF实例转换成关系元组。为此目的，变得必要的是确定RDF中的每一个资源的类，该任务可以通过使用RDF中的可用元性质而促进。

[0071] 因而，当被呈现以外部RDF数据时，转换过程(其示例在下文相对于图4进一步详细描述)通过首先扫描资源以标识指示这样的分类的元性质的出现来尝试分类其中的资源。这些已知的元性质在下文单独地讨论。

[0072] 第一RDF元性质是rdf:type，其形式上定义为：

[0073] *“rdf:type是用于声明资源为类的实例的rdf:Property的实例。*

[0074] *A triple of the form:*

[0075] *R rdf:type C*

[0076] *声明C是rdfs:Class的实例并且R是C的实例。”*

[0077] 因而，一旦转换过程找到用于给定资源的该元性质，则其明确地知晓该资源的类。

+

[0078] 第二RDF元性质是rdfs:domain，其形式上定义为：

[0079] *“rdfs:domain是用于声明具有给定性质的任何资源是一个或多个类的实例的rdf:Property的实例。*

[0080] *A triple of the form:*

[0081] *P rdfs:domain C*

[0082] *声明P是类rdf:Property的实例，C是类rdfs:Class的实例并且由其谓语为P的三元组的主语标注的资源是类C的实例。*

[0083] *在性质P具有多于一个rdfs:domain性质的情况下，则由具有谓语P的三元组的主语标注的资源是由rdfs:domain性质声明的所有类的实例。*”

[0084] 以另一方式声明，该元性质告诉你们rdfs:domain三元组的主语是宾语的性质，并且具有具有如其谓语的该性质的任何其它三元组的主语必然属于该类。因而，考虑RDF在以下表9中阐述的声明。

[0085]

<author> <rdfs:domain> <book>.

<lord of the rings> <author> <J.R.R.Tolkien>.

[0086] 表9。

[0087] 从这些声明，人们知晓“author”是类“books”的性质。当“author”性质被用作针对主语<lord of the rings>的谓语时，人们可以推理出<lord of the rings>属于类“books”。如本领域中已知的，这样的推理可以使用RDFS(RDF模式)推理引擎来标识。

[0088] 第三RDF元性质是rdfs:range，其基本上类似于rdfs:domain，除结果所得的推理应用于三元组声明中的宾语而不是主语。因而，考虑在以下表10中阐述的RDF声明。

[0089]

<eat> <rdfs:range> <food>.

<human> <eat> <vegetables>.

[0090] 表10。

[0091] 从这些声明，人们知晓“eat”是类“food”的性质。当“eat”性质被用作针对宾语“vegetables”的谓语时，人们可以推理“vegetables”属于类“food”。再次，如本领域中已知的，这样的推理可以使用RDFS推理引擎来标识。

[0092] 第四个RDF元性质是rdfs:subClassOf。因而，如果人们遇到形式为<A> <rdfs:subClassOf> <B>的声明，则人们知晓“A”是类并且“A”共享类“B”的所有性质。

[0093] 附加地，应当指出的是，也可以利用涉及类的性质的现有知识。也就是说，如果给定资源不具有告诉其类的任何本体论信息(这是非常常见的)，则转换过程可以标识任何可用的性质并且将那些性质与现有类/表格相比较并且尝试匹配它们，如果可能的话。

[0094] 参照图4进一步图示了图示依赖于上述元性质的转换过程的示例。具体地，图4更详细地图示了三元组储存数据库308和相关数据库304的组件，特别是数据转换中所涉及的那些组件。如所示出的，RDF数据由RDF DBMS 402维持，并且同样地，相关数据由相关DBMS 404维持。在实施例中，来自外部RDF数据储存406的RDF数据可以经由RDF加载器408引入到RDF DBMS 404中，如本领域中已知的。为了实现外部RDF数据向相关数据的转换，三元组储存数据库308可以包括转换桥412和推理引擎414。共同地，转换桥412和推理引擎414构成RDFS转换器，其执行RDF数据410到相关数据416的实际转换。也就是说，如下文更详细描述的，转换桥412检查RDF数据410以标识其中的元性质，并且在需要的情况下在推理引擎414的帮助下，确定可以用于扩展根据相关数据库存储格式所构造的相关数据416的性质。

[0095] 具体地，转换桥412循环访问RDF数据410中的三元组，从而搜索涉及每一个三元组

的主语和宾语的元性质。因而，对于其中找到元性质rdf:type的每一个声明，转换桥412首先提取标识资源的类的宾语。在此之后，转换桥412进行所有表格的搜索以标识与所提取的类名称具有相同的表格名称的表格。如果找到这样的表格，则转换桥412将新资源的性质与现有表格的性质(即列定义)相比较。如果他们不匹配，则转换桥412将新资源的性质添加到表格列定义，即其扩展表格列定义以包括新资源的性质。如果没有找到这样的表格，则转换桥412搜索涉及RDF数据中的资源类的rdfs:domain和rdfs:range元性质，从而尝试确定类的属性。附加地，转换桥412搜索类的宾语的性质。如果在这些另外的努力之后，没有找到这样的性质或属性，则创建新表格，从新资源的名称取得其表格名称，接着是字符串“_UNKNOWN_CLASS”。

[0096] 如果找到元性质rdfs:subClassOf，则转换桥412知晓该资源是类，并且因而其应当被表示为表格。对于该当前类及其父类二者，转换桥412搜索以确定任一个类是否已经具有与其相关联的任何性质。如果具有rdf:type的资源和作为宾语的任一个类被找到，则与该资源相关联的所有性质被提取为另一类的性质。如果找到性质，其中元性质rdfs:domain或rdfs:range作为性质并且任一个类作为宾语，则使用推理引擎414将该性质提取为对应类的性质。如果找到当前或父类中的任一个具有rdfs:subClassOf性质，则这些步骤基于那些子/父类而重复。附加地，对于当前类，转换桥412搜索所有表格以标识与当前类的名称具有相同的表格名称的表格。如果找到这样的表格，则转换桥412将新资源的性质与现有表格的性质(即列定义)相比较。如果它们不匹配，则转换桥412将新资源的性质添加到表格的列定义。然而，如果没有找到这样的表格，则基于当前类名称创建新表格并且之前针对该当前类所收集的性质被用作列定义。如果找到更多的rdfs:subClassOf声明，则基于新的当前类和父类而重复之前的步骤。

[0097] 随着其循环访问RDF数据410，转换桥412可以确定给定资源不具有与其相关联的本体论信息(如由之前描述的元性质所提供的)。在该实例中，转换桥412将尝试基于针对资源的任何已知性质的比较来对资源分类。具体地，转换桥412可以被提供有置信度水平c(其中0 <= c <= 1)。例如，置信度水平可以由工作站206的用户、管理员等提供。不管置信度水平的源如何，转换桥412通过当前用户能够访问的所有可用表格进行搜索，并且对于每一个表格，对列数目计数并且将该列计数值与未经分类的资源的性质数目即性质计数值相比较。将列计数值和性质计数值中的较大者视为n并且较小者视为m，两者之间的共同性质数目p被计数。如果p >= m*c，指示该表格的列与资源的性质之间的相似性足够高，则转换桥412临时将该表格的名称记录在列表中。在已经以此方式处理所有表格之后，搜索列表，并且如果列表为空(指示没有足够类似的表格被标识)，则未经分类的资源不能通过任何已知信息进行分类。在该情况下，转换桥412将未经分类的资源视为新类并且在其后跟着字符串“_UNKNOWN_CLASS”的未知资源的名称之后创建新表格并且将资源插入到新表格中。另一方面，如果列表不是空的，则标识具有最大值p的表格。转换桥412然后假定所标识的表格是资源的类并且比较性质(如以上描述的)，并且扩展表格列定义，如果必要的话。在此之后，将资源插入到该表格中。以此方式，当RDF数据410不包含本体论信息(元性质)并且所有资源共享完全不同的性质时，发生最差情形场景。在该最差情形场景中，然后，转换桥412将生成潜在地大量表格，其中在每一个表格中仅具有一个记录。为了避免该问题，置信度水平可以被设定为0，使得所有未经分类的资源被视为具有相同类，并且因而插入在相同表格中，这

同样地可能不是合期望的结果。因而，置信度水平平衡所创建的表格数目对分类精度。

[0098] 一旦已经完成RDF数据410到相关数据416的转换，则RDF数据416可以添加到相关DBMS 404。在与RDF加载器408相同的脉络中，相关DBMS 404可以与RDF输出器418通信，如本领域中已知的，该RDF输出器418能够将相关数据直接输出到RDF数据420中(例如，如上文描述的)。

[0099] 图4图示了可以与RDF DBMS 402和相关DBMS 404结合地使用的附加组件。例如，如本领域中所已知的，管控组件422可以被提供以如所图示的那样管理每一个用户具有的具体权利(用户权限)、有效用户的标识(用户)和具体用户角色的标识(角色)。如进一步示出的，可以提供许多查询接口以向用户供应各种方式来访问RDF和相关数据。例如，如本领域中已知的，SPARQL端点424支持所谓的SPARQL RDF查询协议426。以此方式，用户可以使用SPARQL查询428直接访问RDF DBMS 404。可替换地，以上指出的统一API430可以用于不仅支持SPARQL查询428和SQL类查询432以用于访问RDF DBMS 402，而且还支持使用SQL查询433以用于访问相关DBMS 402。

[0100] 再次参照图3，以上描述的类型的对象310集中地图示在控制器302内以强调由控制器302采用的对象中心方案。此外，控制器提供起源于对象的使用的许多功能。如表1中所图示的，对象包括一个或多个状态指示符，其可以采用许多值以反映不同状态，这依赖于对象的性质。状态管理组件312针对系统300中的每一个对象追踪这样的状态信息。例如，如下文更详细描述的，单独的对象可以具有与彼此的各种各样的关系，该关系可以反映在(多个)状态指示符中。例如，代表具体数据的对象可以包括对象是驱动另一数据对象(例如如在其中“单价”数据对象将驱动“总购入价”数据对象的情况中)还是由另一数据对象驱动(例如相同示例，但是从“总购入价”数据对象的角度来看)的指示符。可替换地，如本文所使用的，窗口小部件是指本身可以是具有与彼此的各种关系的其它对象(或窗口小部件)的类集的对象。组成对象(和/或其它窗口小部件)之间的这些关系可以反映在许多状态值中，诸如但不限于“包含”、“有子”、“有父”等。此外，状态数据可以反映对象的临时使用状态，例如“可以使用”、“使用”或“已经使用”状态值。又进一步地，状态指示符性质上可以是二进制的，如在“隐藏”对“可见”状态值或者“启用”对“禁用”状态值的情况中。再次，以上示例仅仅是可以采用的众多可能的状态指示符和值的例子。

[0101] 历史管理组件314操作成维持涉及针对每一个对象的修订的信息并且追踪哪个修订是最当前的。类似于以上描述的状态指示符，修订状态可以包括如本领域中所已知的“当前”、“存档”、“删除”或“历史”，其全部都由历史管理组件314针对每一个对象(在数据库304-308内)进行追踪。

[0102] 提供断开组件316以管理在与控制器302的连接丢失的事件中可能随着某些对象而出现的冲突情况。如下文更详细描述的，由控制器302追踪的对象(尤其是涉及软件窗口小部件或其它分立功能组件的那些)可以用于构造最终用户应用。为此目的，当基于某些对象构建应用时，查询应用的作者以指明某些对象甚至在断开事件中也是可用的，并且该信息由断开组件316追踪。然后经由最终用户应用服务器使应用对于最终用户可用。当最终用户访问最终用户应用服务器上的应用时，服务器与客户端处理设备(例如桌上型计算机、膝上型计算机、移动无线设备等)协商以确定多少本地存储可用于分配给断开功能，其中本地存储的期望量部分地依赖于即便断开也要求可用的特定对象。该与客户端处理设备协商的

过程可以使用相同应用针对许多其它最终用户处理设备重复，使得每一个客户端处理设备包括用于指定对象的相同本地存储。当相对于一个最终用户的客户端处理设备发生断开时，控制器302使用已知技术检测该条件，并且将该事实通过断开组件316告知给其它最终用户客户端设备。附加地，断开的最终用户客户端设备切换到其中它使用它的本地存储来维持指定对象的操作的模式。例如，如果指定对象是追踪购买订单的放置的窗口小部件，则该窗口小部件的任何使用(例如“针对1,000个零件从公司A向公司B发送P.O.”仅被维持(在继续能够创建、读取、更新和删除数据的意义下)在本地存储中。同时，其它最终用户客户端设备可以继续正常地操作，包括以可能与断开的客户端设备冲突的方式使用相同指定对象，例如“针对2,000个零件从公司A向公司B发送P.O.”。然而，当其这样做时，断开组件316通过其它最终用户客户端追踪指定组件的使用。当断开的客户端设备恢复与控制器302的连接时，存储在其本地存储中的数据上载到控制器302并且断开组件316可以检测冲突的发生。实质上，断开组件316“孤立”涉及由断开的最终用户客户端在其断开时段期间所使用的任何指定组件的任何数据。在检测到冲突时，断开组件316可以以不同方式解决该冲突。因而，在实施例中，断开组件316可以具有关于各种最终用户客户端设备的分级的规则。例如，在企业或类似分级组织的实体内，具体最终用户客户端可以与标题、位置或其它优先指示符相关联以确定哪个最终用户客户端应当优先于另一个，并且因此依照由具有较高优先级的最终用户客户端提供的数据来自动解决冲突。在其中这样的自动解决不大可能的那些实例中，控制器302可以利用请求将冲突数据发送给冲突的最终用户客户端设备以解决冲突。在此之后，假定冲突客户端能够解决冲突，则数据可以被提供回到断开组件316，从而指示可以如何解决冲突，即要存储哪些数据。

[0103] 基于由状态管理组件312维持的状态信息，可以构造用于每一个对象的“社交”网络。也就是说，使用针对每一个对象所维护的关系信息，有可能创建对象的网络表示及其与彼此的关系。例如，“雇员名”对象和“雇员姓”对象可以均反映相对于“雇员姓名”对象的“所属”状态，“雇员姓名”对象继而可以具有其自身到其它对象的连接等等。这样的网络可以由网络组件318使用已知网络发现技术导出。例如，使用如例如由数据挖掘服务器222(用于实现下文描述的自动数据挖掘组件332)所提供的已知数据挖掘技术(例如根本原因分析、分类、集群、关联规则发现和/或回归分析)。此外，可以提供根本原因分析组件320(不要与由网络组件318用来产生对象社交网络的根本原因分析混淆)，如所示出的。再次使用已知技术(诸如神经网络分析或回归分析)，对象社交网络(如由网络组件318所提供的)内的所谓的根本原因可以相对于某些对象而标识。为了更准确，根本原因是这样的社交网络不能总是直接标识，并且替代地，有时标识作为潜在的因果关系的相关关系。也就是说，对于相对简单且清楚的社交网络，根本原因可以确定地标识。然而，复杂和/或不清楚的社交网络、相关关系可以被标识经受附加人类分析。例如，涉及对象“雇员效率”的许多对象可以包括“雇员年龄”、“雇员技能水平”、“星期几”、“工厂温度”等。在神经网络分析的情况下，这些对象之下的数据可以使用已知技术来分析以揭示网络功能，其有效地揭示在预测“雇员效率”对象的值方面最显著因素。这样的根本原因的标识然后可以用于创建之前不存在的对象之间的关联，或者更新或甚至删除之前定义的关联。

[0104] 如上文提及的，用于与存储在系统200、300中的数据结合地使用的应用可以使用多个分级用户接口来开发。在所图示的示例中，分级用户接口包括第一主要开发者接口

322、第二主要开发者接口324和次要开发者接口326。要指出的是，所有开发者接口322-326是可选的并且可以提供它们的任何组合。一般地，每一个开发者接口322-326具有两个使用模式或角色：作为可以由不同用户针对不同目的单独地使用的独立平台，或者作为与(多个)其它平台协同的相关平台(如果提供的话)以充当一个统一系统。在实施例中，第一主要开发者接口322、第二主要开发者接口324和次要开发者接口326用作软件开发中的接连地较高抽象层；抽象层越高，其使用起来就越简单，因为对于应用开发而言随着逐渐更多的编程细节被隐藏。

[0105] 因而，在实施例中，第一主要开发者接口322是集成开发环境(IDE)，诸如如本领域中已知的Apache Eclipse。使用第一主要开发者接口322，相对技术熟练的程序员可以使用它来开发任何类型的软件。第二主要开发者接口324可以使用任何数目的GUI应用构建器来实现，包括用于实现主要开发者接口322的相同应用，其可以用于以中间抽象水平构造完整功能GUI应用。次要开发者接口326可以包括任何数目的图形、web应用构建器，诸如如本领域中已知的Zoho Creator，其可以用于允许基本上没有软件开发技能的人员基于高水平功能构建块来构造应用。因而，由第一主要开发者接口322提供的低水平抽象是显然的，因为其用户应对具体编程语言特征，而在第二主要开发者接口324中使用的功能是编程语言独立的，并且在次要开发者接口326中，根本不存在编程特定术语或特征。

[0106] 在操作中，如本领域中所已知的，第一主要开发者接口322提供准许其用户生成和修改软件代码的许多模式。例如，一些IDE装备有定义的可选任务。当选择给定任务时，代码模板还可以被选择使得IDE基于所选模板自动地生成代码。可替换地，用户可以通过一系列下拉菜单来定义操作，该菜单连续更新以示出可用操作。当用户选择各种操作时，代码被自动生成。在又另一实施例中，自然语言处理引擎可以用于解析由用户提供的自然语言文本以提供中间命令声明，其然后可以被分析以自动提供所生成的代码。在所有实例中，自动生成的代码可以由用户如所期望的那样进行修改以提供最终期望的代码。

[0107] 如本领域中已知的，第二主要开发者接口324提供“拖放”图形用户接口，其中在工具框中提供各种用户接口控件。各种可用控件可以被拖拽到设计区域以创建所选控件的实例，该实例可以随后被选择和配置成展现某些行为。类似地，可以将任何期望的事件定义、流控制或动作添加到所选控件实例。通过将这样的控件组合在一起，可以产生窗口小部件或更完整的应用，从而实现期望的用户接口功能。一旦完整地配置，则结果所得的窗口小部件或应用可以被发布。

[0108] 要指出的是，由第一和第二主要开发者工具322、324产生的任何代码和/或窗口小部件可以作为对象由控制器302存储。

[0109] 类似于第二主要开发者接口324，次要开发者接口326也基于“拖放”GUI。然而，针对次要开发者接口326所提供的工具框可以包括发布的窗口小部件或应用，其可以被选择和组合在设计区域中。一旦定义了完整应用，则第二主要开发者接口326使用已知技术来生成应用元文件，其描述各个窗口小部件的操作及其与彼此的相应关系，例如使用用户接口标记语言(诸如Qt元语言(QML))和/或功能标记语言(诸如行为标记语言(BML))。结果所得的应用元文件然后传递给代码生成器328，其生成源和可执行代码。这样的代码生成器的示例是从Eclipse Foundation可获得的Acceleo开放源代码生成器。结果所得的源代码和可执行代码可以作为对象由控制器302存储，并且可执行代码330可以经由适合的应用服务器

等而对最终用户做成可用。

[0110] 如上文指出的，接口322-326的每一个也可以以协同方式使用。例如，第一主要开发者接口322可以用于关注于使用与它相兼容的特定编程语言的开发构造，即构建编程语言实体和逻辑包装体以供第二主要开发者工具324使用。例如，使用第一主要开发者接口322，开发者可以将Java GUI组件(比如文本输入框)包装成具体对象并且使该对象可用于第二主要开发者接口324(通过控制器302)，由此准许第二主要开发者接口324将该对象添加到工具框中以供随后使用。以该方式，第一主要开发接口322可以被视为针对第二主要开发者接口324的“插件”，由此扩展第二主要开发者接口的功能。

[0111] 继而，第二主要开发者接口324可以在关注于其可以开发的应用的类型方面协同地使用，即构建GUI组件和逻辑包装体以供次要开发者接口326使用。例如，使用第二主要开发者接口324，开发者可以包装“提交”按钮以包括使得按钮上的单击能够引起当前屏幕上的所有数据被聚集并提交至数据库304-306的逻辑，并且将该对象馈送给次要开发者接口326，由此准许次要开发者接口326将该对象添加到其工具框以供随后使用。再次，以此方式，第二主要开发接口324可以被视为针对次要开发者接口326的“插件”，由此扩展次要开发者接口的功能。

[0112] 再次参照图3，系统300包括增强用户与所存储的数据进行交互的能力的各种功能。在一个实施例中，自动数据挖掘组件332实现各种已知数据挖掘算法，其可以针对数据库304-306(如由控制器302所调解的)中所存储的数据应用。在特定实施例中，自动数据挖掘组件332操作成针对给定数据挖掘任务最佳地预处理数据，并且选择用于数据挖掘任务的最佳数据挖掘算法。

[0113] 如本领域中已知的，数据挖掘在对要分析的数据执行预处理时产生最佳结果。然而，这样的预处理可以强烈地依赖于要分析的数据的性质。自动数据挖掘组件332可以参与训练以便自动选择最佳数据预处理。为此目的，首先聚集采样数据集并且提取其统计特性。这样的统计特性可以包括例如数学特征，诸如平均、模、中值、范围和标准偏差等。它们还可以包括简单事实，诸如属性数目、每一个数学的类型(例如标定对数值)、数据集大小等。由此表征数据集之后，可以针对数据集运行数目为N的已知数据预处理算法，使得单独地存储针对每一预处理算法的结果所得的经预处理的数据。在此之后，数目为M的已知数据挖掘算法可以在每一个经预处理的数据集上运行，由此产生NxM数据挖掘结果集。每一个数据挖掘结果集然后使用已知技术来评估，以评定相关预处理和数据挖掘算法组合的结果所得的精度和准确度。

[0114] 在可能的情况下，用于每一个数据预处理算法的参数还可以变化以标识预处理算法和参数以及数据挖掘算法的最佳组合。一旦被标识，则预处理算法/参数/数据挖掘算法的最佳组合可以被指定为类属性并且数据集的统计特性可以被指定为输入属性。这些类/输入属性然后用于使预处理选择学习模型递增，使得具有基本上匹配的统计特性的随后数据集可以以相同方式来预处理。

[0115] 附加地，情况可以是某些数据挖掘算法对于给定数据挖掘任务而言比其它更好。以类似于以上所描述的用于训练以选择最佳预处理算法的方式，自动数据挖掘组件332还可以参与训练以便基于要执行的具体数据挖掘任务而自动选择最佳数据挖掘技术。为此目的，再次聚集采样数据集并且提取其统计特性。由此表征数据集之后，可以针对数据集运行

数目为N的已知数据预处理算法，使得单独地存储用于每一数据挖掘算法的结果所得的数据集。每一个数据挖掘结果集然后使用已知技术来评估以评定每一数据挖掘算法的结果所得的精度和准确度。在可能的情况下，用于每一数据挖掘算法的参数也可以变化以标识数据挖掘算法和参数的最佳组合。一旦被标识，则数据挖掘算法和参数的最佳组合可以被指定为类属性并且数据集的统计特性可以被指定为输入属性。这些类/输入属性然后用于使数据挖掘选择学习模型递增，使得要用于经受给定数据挖掘任务并具有基本上匹配的统计特性的随后数据集可以以相同方式来处理。

[0116] 在实施例中，最佳预处理和/或数据挖掘算法的益处可以通过另外的过程获得。在该过程中，要预处理或经受给定数据挖掘任务的数据集可以再次统计地表征，如上文描述的。基于结果所得的统计特性，最佳的k个预处理或数据挖掘算法基于数据集和输入属性的统计特性之间的相似性程度来选择，如上文描述的。并行地，如本领域中已知的，输入数据集可以经受数据减少，以使得所有可用预处理或数据挖掘算法可以针对经减少的输入数据集应用，并且选择最佳的n个预处理或数据挖掘算法。在又另一并行路径中，机器学习模型可以用于确定最佳的m个预处理或数据挖掘算法。在此之后，比较k、m和n个不同的预处理或数据挖掘算法的结果以选择最佳的h个预处理或数据挖掘算法。这些h个预处理或数据挖掘算法然后针对输入数据集运行并且结果然后一起平均。结果所得的平均输出然后应当表示预处理或数据挖掘算法的最佳可能组合。

[0117] 在另一数据挖掘实施例中，提供了可选地与自然语言处理引擎336结合地操作的问题驱动的数据挖掘组件334。问题驱动的数据挖掘组件334为具有很少数据挖掘经验或者没有数据挖掘经验的用户提供执行数据挖掘任务的机构。最终用户可以将数据挖掘请求提供给控制器302，其然后可以将请求直接地提交至问题驱动的数据挖掘组件334，或者在请求以自然语言文本表述的情况下，通过NLP引擎336将请求提交至问题驱动的数据挖掘组件334以用于转换成可以由问题驱动的数据挖掘组件334用来分析必要数据集的指令。

[0118] 具体地，问题驱动的数据挖掘组件334经由例如用于该具体目的的用户接口接收以自然语言表述的用户的问题。当接收到这些复杂问题(例如以“为什么”或“如何”的形式表述的问题)时，问题驱动的数据挖掘组件334通过NLP引擎组件336调用处理(如下文描述的)。在NLP引擎组件336不能处理问题的复杂形成的事件中，其将把问题解析成可以由自动数据挖掘组件332实现的数据挖掘任务。NLP引擎组件336将针对数据挖掘操作的需要通知给问题驱动的数据挖掘组件334，这使得问题驱动的数据挖掘组件334生成发起数据挖掘任务所要求的参数(例如，以根据下文描述的API协议表述并且在随附附录中阐述的请求的形式)。这些参数然后用于发起如由自动数据挖掘组件332所执行的数据挖掘任务，其中结果被返回给问题驱动的数据挖掘组件334。为了向用户提供结果，然后问题驱动的数据挖掘组件334然后将结果传递给NLP引擎组件336。

[0119] 在实施例中，如上文所指出的，为了提供可用的数据挖掘操作，自动数据挖掘组件332可以暴露API方法以接收以HTTP(超文本传输协议)POST请求的格式的外部请求，其要求执行具体类型的数据挖掘操作。响应于请求，API可以以另一HTTP POST格式返回完成所请求的操作的估计时间。如本领域中已知的，并且如上文指出的，可以提供任何各种不同类型的数据挖掘任务，包括但不限于根本原因分析、分类、集群、关联规则发现、回归分析等。

[0120] 作为总结，通过API的处理可以描述如下：

[0121] 1．API方法接收HTTP POST请求。

[0122] 2．API方法提取请求数据并且解析数据。

[0123] 3．API方法验证请求数据。如果请求有效，则处理在步骤5处继续。

[0124] 4．如果请求无效，则API方法返回包含错误信息的HTTP POST响应并且处理终止。

[0125] 5．当请求有效时，API方法调用时间估计模块，其基于所选择的数据计算执行请求所要求的时间的估计。

[0126] 6．API方法返回包含估计时间的HTTP POST响应。

[0127] 7．基于请求中的信息，API方法经由控制器标识暗含的对象，由此标识所要求的数据、自动应用数据预处理步骤并且选择最佳算法(如上文描述的)，并且运行数据挖掘过程。

[0128] 8．当过程完成时，API方法将结果返回给请求者。

[0129] 在目前优选的实施例中，经由基于HTTP的接口所发送的消息使用JSON标准格式。关于API方法的另外细节在此之后提供在附录中。

[0130] 如上文所总结的，由自动数据挖掘组件332暴露的API针对所要求的报头和附录中所定义的JSON模式来验证每一个POST请求，其中在POST响应中返回任何结果所得的错误消息。在实施例中，自动数据挖掘组件332应当以预定义的格式(诸如csv或arff文件格式)接受数据集上载并且为每一个上载的数据集提供唯一标识。附加地，自动数据挖掘组件332可以为最终用户提供一个或多个数据集输出器助手工具(如本领域中已知的)，以帮助他们从其现有数据库向可接受格式输出数据。

[0131] 如上文相对于自动数据挖掘组件332所描述的，问题驱动的数据挖掘组件334还可以自动选择最佳数据预处理和/或数据挖掘算法。为了提供针对所请求的数据挖掘任务的时间估计，问题驱动的数据挖掘组件334可以经由由自动数据挖掘组件332所暴露的API而获得时间估计。这样的估计可以基于输入数据的特性来计算，包括大小、所选择的数据准备方法、所选择的训练模式等、以及当前分配用于该任务的计算资源。这通过首先针对学习基础数据集使用机器学习算法来完成，该学习基础数据集在大小方面足够大并且在数据特性方面足够多样以最佳地反映一般数据挖掘任务特性。在对该数据集学习之后，问题驱动的数据挖掘组件334将开发可以用于时间估计的模型。对时间估计学习模型的改进可以通过遵循初始训练和部署的所有用户输入的类集来提供；通过周期性地重新运行机器学习算法，模型的精度可以持续增加。

[0132] 如上文指出的，问题驱动的数据挖掘组件334可以接受以自然语言表述的请求，该请求可以由NLP引擎336进一步处理，这提供两个主要功能：到数据库的自然语言接口(NLIDB)和自然语言应用生成(NLAG)，如下文详细描述的。

[0133] NLIDB功能允许最终用户以自然(即人类可理解的)语言提交查询。例如，这样的查询通常包括比如“什么”、“谁”、“如何”等的表述，例如“哪个州具有我们产品的最高销售额?”以及“谁在去年挣得多于$10,000.00?”。NLP引擎336中的NLIDB模块解析自然语言问题并且将它们翻译成更技术性的查询语言，诸如SQL等，或者优选地翻译成上文描述的统一SQL类API，其继而被翻译成底层数据引擎的本机查询API。

[0134] NLIDB模块采取“逆”方案以解析自然语言问题。也就是说，其不使用统计解析器来解析用户的输入问题，因为这样的技术频繁地引起不准确的结果。相反，NLIDB模块系统简

单地将一些预处理之后的用户的输入映射到预定义的问题/回答表格(Q&A表格)中的可用问题,该表格包括所有“支持的”问题及其对应回答。当实现时,包括在该Q&A表格中的“回答”实际上是可以用于获得回答相关联的问题的数据的查询。Q&A表格基于存储在数据库中的可用模式和数据而生成。通过应用自然语言语法规则,NLIDB模块生成所有可能的问题,其具有包括相同问题的不同形式的明确回答。该策略牺牲相对更便宜的存储容量(存储该巨大的列表所需要的),以得到解析准确性和实时执行。由于解析如匹配字符串那样简单,所以执行非常快速并且实现实时响应。

[0135] 如果用户的输入不与任何所支持的问题匹配,则如本领域中已知的统计解析(SP)过程被用于做出寻找回答方面的最佳努力。SP过程首先从问题中消除停用词,从而仅留下关键词。SP过程然后使用关键词来执行在数据库中的文本搜索并且返回相关数据对象。然而,该过程不保证将找到正确回答。其做出尝试得到针对问题的正确或相关回答的最佳努力,并且可能返回难以理解的总体上不相关的回答或未经格式化的数据。在执行SP过程之后,为用户提供潜在回答的列表和对回答评分的请求,即参与主动学习。如果用户在所返回的结果中找到正确回答,则请求他/她对该回答给出良好的评分,这是可能像竖起拇指那样简单的事情。如果用户以其它方式对任何回答都不满意,则请求他/她给出差的评分,这是可能像拇指朝向那样简单的事情。如果用户未对回答评分,则评分被视为“中立的”。记录每一个和每一次用户输入。对于与所支持的问题不匹配并且因而由SP处理的问题,存在设计成存储对应记录的贮藏库。专家团队可以分析这些记录,并且针对最常见的误处理输入,将问题添加到所支持的问题并且更新Q&A表格。

[0136] 如上文所提及的,基于存储在一个或多个数据库中的数据的所有可回答的问题可以通过底层数据库模式的知识来标识。继而,模式字段由最终用户定义。尽管模式字段通常以有意义的词定义,但是不存在将不会使用非自然语言词/符号(诸如数字、代码或无意义的字符)的保证。对于具有非自然语言符号的模式字段,NLIDB模块首先尝试从数据类型定义模式字段的语义意义。如果数据类型不可用或者不满足需要,则NLIDB模块请求用户定义其语义意义。例如,这可以经由上文描述的次要开发者接口326完成。

[0137] 对于可解释的模式字段名称,NLIDB模块查找本体论定义中的词,即在底层本体论中所使用的结构的描述。一旦定位了意义,则NLIDB模块开始扩展可以用作用户查询中的词的可替换方案的别名(alias)的列表。该扩展可以以许多不同方式执行。根据一种方法,上层本体论定义被用作别名。例如,“雇员”是与“人”相同的事物。根据另一方法,词典可以用于标识已知同义词。相反地,根据另一方法,可以使用反义词的辞典来针对给定词标识反义词。该方法对于动词尤其有用,因为动词可以用作否定前缀及其反义词的组合,例如“破损”和“不运转”可以是指相同意义。使用这些方法的组合,NLIDB模块可以构建针对模式中的具体词的别名列表。此外,为了使用以上指出的技术扩展可用于别名标识的词的数目,可能合期望的是解决缩写的意义。例如,“P.O.”可以意指许多事情,但是在购买部应用中,其可能意指“购买订单”并且将在包括该上下文的缩写定义的列表中如此定义。如果上下文不充分,则可以通过为用户呈现可用选项的列表而实现解疑。

[0138] 在发现模式词及其别名之后,NLIDB模块开始基于模式词及其关系来对潜在问题进行汇编。为此目的,其使用本体论信息和自然语言句法二者。词的本体论信息可以直接映射到问题词。例如,如果诸如“DOB”之类的模式词具有类型“时间”,则应当生成问题“何

时...?”。模式字段的彼此的关系是用于生成问题的另一重要基础。例如,如果雇员数据对象包含“名称”字段和“DOB”字段,则可以生成问题“John Doe的生日是什么?”或“John Doe何时出生?”。附加地,除将字段名称映射到问题词之外,NLIDB模块还将它们映射到命令词,诸如“向我示出”、“我需要知道”、“给我”等。该映射生成不开始于问题词的“问题”。

[0139] 在生成问题之后,相应地生成其对应查询。例如,问题“John Doe的生日是什么?”具有对应的SQL查询“SELECT DOB FROM Employee WHERE Name = ‘John Doe’”。该查询用作“回答”并且连同自然语言问题一起存储在Q&A表格中。

[0140] 在使用中,NLP引擎336允许最终用户通过例如文本框录入问题。自动完成可以用于建议什么问题可用。如果用户键入连同之前键入的词一起不与Q&A表格中的任何可用问题匹配的词,则自动完成将示出空列表以警告用户已经输入了潜在不支持的问题。使用拼写检查服务逐词检查用户的输入。如果标识具有打字错误的词,则其可以以某种方式高亮,例如通过以彩色示出它。用户可以通过使用一个所建议的词来修正它,或者不管它。如果用户键入不遵循正式自然语言句法(英语语法)的问题,则可以允许用户完成键入并且然后为其提供类似于用户输入但是在句法上正确的建议问题的列表。

[0141] 如果用户输入确实与可用问题匹配,则NLIDB模块搜索Q&A表格中的问题,定位以数据库查询形式存储的“回答”,针对数据库执行查询,并且然后将结果返回给最终用户。如果用户输入不与可用问题匹配,则采用如上文描述的统计处理。

[0142] 关于NLAG功能,采用上文相对于NLIDB功能所描述的相同方法论,除了模式字段由应用模块关键词替换以及问题由功能描述声明替换。也就是说,NLAG功能帮助用户(例如次要开发者接口用户等)基于自然语言描述生成应用。应用通过功能模块或组件组装,其中每一个模块实现子功能。应用的描述应当解释应用的预期功能或者应当实现什么应用。示例包括“我需要管理我的雇员的程序”或更具体地比如“我想要我可以从其添加、编辑、更新和删除雇员信息、接受P. O.并且查看组装线状态的应用”。这些描述揭示高水平或分级功能要求。

[0143] 通过利用本体论词典,NLP引擎336内的NLAG模块识别不同水平的要求。为了支持该功能,应用模块(例如如上文描述的窗口小部件)的作者必须以动词-名词模式的格式提供模块的功能的描述。例如,雇员管理模块可以具有描述“管理雇员”而组装线仪表盘模块可以具有描述“提供组装线状态”。然后在本体论词典中查找这些动词-名词对,并且执行上文关于NLIDB功能所描述的相同过程,包括别名扩展、问题(在该情况下是声明)生成和查询(在该情况下是模块组装)生成。问题(声明)解析阶段在通过不匹配输入的统计处理和自动完成来限制用户输入方面也是类似的。在用户的输入已经成功解析并且返回模块列表之后,适用的开发工具(例如次要开发者接口326)允许用户将模块组装成统一应用,如上文描述的。

[0144] 最后,提供了报告引擎组件340。在实施例中,报告引擎组件是次要开发者接口326的子组件。具体地,其是GUI报告构建器,其允许用户通过首先生成包含系统中的所有(所选)数据的大表格来构建报告。从该大表格,用户可以基于现有列上的计算而移除列、向列添加聚合功能(例如求和、平均等)、或者添加新列,从而得到新表格。该过程可以重复直到获得最终期望的表格。在设立该表格之后,用户可以在一个屏幕中查看所有表格,并且报告引擎组件340可视化表格列之间的关系。附加地,用户可以建立报告更新频率,使得报告引

擎组件340不必在每次更新构成数据元素时执行更新。

[0145] 尽管已经示出并且描述了具体的优选实施例，但是本领域技术人员将认识到，可以做出改变和修改而不脱离本教导。因此预期到，上述教导的任何以及全部修改、变化或等同方案都落在以上公开并且在本文要求保护的基本底层原理的范围内。

[0146] 附录

[0147] 1. 数据上载API

[0148] URL

[0149] https://www.beulahworks.com/dm/v1/data_upload

[0150] POST 请求要求的字段

[0151] POST /dm/v1/data_upload HTTP/1.1

[0152] 内容类型:text/csv

[0153] 内容长度:3876502

[0154] 字符集:utf-8

[0155] 接受-字符集:utf-8

[0156] 主机:www.beulahworks.com:1234 (可配置)

[0157] 文件名:“abc.arff”

[0158] {数据文件}

[0159] 内容类型字段应当具有MIME类型的所有支持数据文件格式，包括

[0160] CSV:text/csv

[0161] ARFF:application/vnd.arff (定制MIME类型;可以在web服务器中设定)

[0162] 数据文件的大小不受限制。其可以在web服务器配置文件中设定。

[0163] POST响应要求的字段

[0164] HTTP/1.1 200 OK

[0165] 内容类型:application/json; charset=utf-8

[0166] {响应JSON}

[0167] 响应JSON模式

[0168] {

```
"type": "object",
"$schema": "http://json-schema.org/draft-03/schema",
"required": true,
"properties": {
  "statusCode": {
    "type": "string",
    "required": true
  },
  "statusDescription": {
    "type": "string",
    "required": true
  },
  "status": {
    "type": "string",
    "required": true,
    "enum": [
      "success",
      "failure"
    ]
  },
  "transactionId": {
    "type": "string",
    "required": true
  },
  "transactionTime": {
    "type": "string",
    "required": true
  },
  "datasetId": {
    "type": "string",
    "required": true
```

[0169]

```
        }
    }
}
```

[0171] 响应JSON示例

[0172]
```
{
    "status": "success",
    "statusCode": "0",
    "statusDescription": "Success",
    "transactionTime": "2013-12-10T03:08:23:63Z",
    "transactionId": "241b9632-ebfb-4be2-9d6d-64910f995182",
    "datasetId": "FBADDC8E-4007-4901-9CBF-328318E83DC5",
}
```

[0173] 其中“datasetId”将用于以下的API方法。

[0174] 2．训练API

[0175] A．分类训练

[0176] URL

[0177] https://www.beulahworks.com/dm/v1/classification_train

[0178] POST 请求要求的字段

[0179] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0180] 请求JSON模式

[0181]
```
{
    "type":"object",
    "$schema": "http://json-schema.org/draft-03/schema",
    "id": "http://jsonschema.net",
    "required":false,
    "properties":{
```

[0182]

```
"algorithm": {
        "type":"array",
        "id": "http://jsonschema.net/algorithm",
        "required":false,
        "items":
                {
                        "type":"object",
                        "id": "http://jsonschema.net/algorithm/0",
                        "required":false,
                        "properties":{
                                "name": {
                                        "type":"string",
                                        "id": "http://jsonschema.net/algorithm/0/name",
                                        "required":false
                                },
                                "options": {
                                        "type":"object",
                                        "id":
"http://jsonschema.net/algorithm/0/options",
                                        "required":false,
                                        "properties":{
                                                "prune": {
                                                        "type":"boolean",
                                                        "id":
"http://jsonschema.net/algorithm/0/options/prune",
                                                        "required":false
                                                }
                                        }
                                }
                        }
                }
```

[0183]

```
},
"classAttributeName": {
        "type":"string",
        "id": "http://jsonschema.net/classAttributeName",
        "required":false
},
"datasetId": {
        "type":"string",
        "id": "http://jsonschema.net/datasetId",
        "required":true
},
"modelName": {
        "type":"string",
        "id": "http://jsonschema.net/modelName",
        "required":true
},
"preprocessor": {
        "type":"array",
        "id": "http://jsonschema.net/preprocessor",
        "required":false,
        "items":
                {
                        "type":"object",
                        "id": "http://jsonschema.net/preprocessor/0",
                        "required":false,
                        "properties":{
                                "name": {
                                        "type":"string",
```

```
                                                    "id":
        "http://jsonschema.net/preprocessor/0/name",
                                                    "required":false
                                                },
                                                "options": {
                                                    "type":"object",
                                                    "id":
        "http://jsonschema.net/preprocessor/0/options",
                                                    "required":false,
                                                    "properties":{
                                                        "removeAttriIndex": {
                                                            "type":"number",
                                                            "id":
        "http://jsonschema.net/preprocessor/0/options/removeAttriIndex",
                                                            "required":false
                                                        }
                                                    },
                                                    "type": {
                                                    "type":"string",
                                                    "id":
        "http://jsonschema.net/preprocessor/0/type",
                                                    "required":false
                                                }
                                            }
                                        }


                    }
                }
        }
```

[0184]

[0185] 请求JSON示例

[0186]
```
{
  "datasetId": "FBADDC8E-4007-4901-9CBF-328318E83DC5",
  "preprocessor": [
    {
      "name": "Remove",
      "type": "filter",
      "options": {
        "removeAttriIndex": 2
      }
    }
  ],
  "algorithm": [
    {
      "name": "J48",
      "options": {
        "prune": false
      }
    }
  ],
  "classAttributeName": "Gender",
  "modelName": "GenderPredictor"
}
```

[0187] 其中“classAttributeName”在数据集被更新为ARFF文件时不是要求的；“algorithm”和“preprocessor”对于自动数据挖掘不是要求的；“algorithm”和“preprocessor”属于阵列类型；其意味着API支持多个专业处理器和算法。如果指定多个预处理器，则它们的所有被应用于数据集；如果指定多个算法，则算法被单独地应用于数据集并且将报告经平均的结果。

[0188] POST 响应要求的字段

[0189] HTTP/1.1 200 OK

[0190] 内容类型：application/json; charset=utf-8

[0191] {响应JSON}

[0192] 响应JSON模式

[0193]

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "required": true,
  "properties": {
    "statusCode": {
      "type": "string",
      "required": true
    },
    "statusDescription": {
      "type": "string",
      "required": true
    },
    "status": {
      "type": "string",
      "required": true,
      "enum": [
        "success",
        "failure"
      ]
    },
    "transactionId": {
      "type": "string",
      "required": true
```

```
        },
        "transactionTime": {
          "type": "string",
          "required": true
        },
[0194]  "jobId": {
          "type": "string",
          "required": true
        }
      }
    }
```

[0195] 响应JSON示例

```
{
  "status": "success",
  "statusCode": "0",
  "statusDescription": "Success",
[0196]
  "transactionTime": "2013-12-10T03:08:23:63Z",
  "transactionId": "241b9632-ebfb-4be2-9d6d-64910f995182",
  "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5"
}
```

[0197] 其中“statusCode”和“statusDescription”是预定义的标准成功/错误消息集；“transaction Time”是在API方法发出响应时的UTC时间；transactionID是将用于录入和划分目的的UUID；jobId将由其它API方法用来检查具体工作的估计时间。

[0198] B．集群训练

[0199] URL

[0200] https://www.beulahworks.com/dm/v1/clustering_train

[0201] POST请求要求的字段

[0202] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0203] 请求JSON模式

[0204]

```
{
        "type":"object",
        "$schema": "http://json-schema.org/draft-03/schema",
        "id": "http://jsonschema.net",
        "required":false,
        "properties":{
                "algorithm": {
                        "type":"array",
                        "id": "http://jsonschema.net/algorithm",
                        "required":false,
                        "items":
                                {
                                        "type":"object",
                                        "id": "http://jsonschema.net/algorithm/0",
                                        "required":false,
                                        "properties":{
                                                "name": {
                                                        "type":"string",
                                                        "id": "http://jsonschema.net/algorithm/0/name",
                                                        "required":false
                                                },
                                                "options": {
                                                        "type":"object",
                                                        "id":
"http://jsonschema.net/algorithm/0/options",
                                                        "required":false,
```

```
                                                    "properties":{
                                                        "numClusters": {
                                                            "type":"number",
                                                            "id":
        "http://jsonschema.net/algorithm/0/options/numClusters",
                                                            "required":false
                                                        }
                                                    }
                                                }
                                            }
                                        }


                    },
                    "datasetId": {
                        "type":"string",
                        "id": "http://jsonschema.net/datasetId",
                        "required":true
                    },
                    "preprocessor": {
                        "type":"array",
                        "id": "http://jsonschema.net/preprocessor",
                        "required":false,
                        "items":
                            {
                                "type":"object",
                                "id": "http://jsonschema.net/preprocessor/0",
                                "required":false,
                                "properties":{
                                    "name": {
                                        "type":"string",
```

[0205]

[0206]

```
                                        "id":
"http://jsonschema.net/preprocessor/0/name",
                                        "required":false
                                    },
                                    "options": {
                                        "type":"object",
                                        "id":
"http://jsonschema.net/preprocessor/0/options",
                                        "required":false,
                                        "properties":{
                                            "removeAttriIndex": {
                                                "type":"number",
                                                "id":
"http://jsonschema.net/preprocessor/0/options/removeAttriIndex",
                                                "required":false
                                            }
                                        },
                                        "type": {
                                            "type":"string",
                                            "id":
"http://jsonschema.net/preprocessor/0/type",
                                            "required":false
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

[0207] 请求JSON示例

```
{
  "datasetId": "FBADDC8E-4007-4901-9CBF-328318E83DC5",
  "preprocessor": [
    {
      "name": "Remove",
      "type": "filter",
      "options": {
        "removeAttriIndex": 2
      }
    }
  ],
  "algorithm": [
    {
      "name": "K-Means",
      "options": {
        "numClusters": 5
      }
    }
  ]
}
```


[0208]

[0209] 响应POST要求的字段

[0210] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0211] 响应JSON模式

[0212] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0213] 响应JSON示例

[0214] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0215] C．关联规则发现训练

[0216] URL

[0217] https://www.beulahworks.com/dm/v1/association_rule_train

[0218] POST请求要求的字段

[0219] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0220] 请求JSON模式

[0221]

```
{
        "type":"object",
        "$schema": "http://json-schema.org/draft-03/schema",
        "id": "http://jsonschema.net",
        "required":false,
        "properties":{
                "algorithm": {
                        "type":"array",
                        "id": "http://jsonschema.net/algorithm",
                        "required":false,
                        "items":
                                {
                                        "type":"object",
                                        "id": "http://jsonschema.net/algorithm/0",
                                        "required":false,
                                        "properties":{
                                                "name": {
                                                        "type":"string",
                                                        "id": "http://jsonschema.net/algorithm/0/name",
                                                        "required":false
                                                },
```

[0222]

```
                                        "options": {
                                            "type":"object",
                                            "id":
"http://jsonschema.net/algorithm/0/options",
                                            "required":false,
                                            "properties":{
                                                "numRules": {
                                                    "type":"number",
                                                    "id":
"http://jsonschema.net/algorithm/0/options/numRules",
                                                    "required":false
                                                }
                                            }
                                        }
                                    }
                                }


                    },
                    "datasetId": {
                        "type":"string",
                        "id": "http://jsonschema.net/datasetId",
                        "required":true
                    },
                    "preprocessor": {
                        "type":"array",
                        "id": "http://jsonschema.net/preprocessor",
                        "required":false,
                        "items":
                            {
                                "type":"object",
```

```
                                    "id": "http://jsonschema.net/preprocessor/0",
                                    "required":false,
                                    "properties":{
                                        "name": {
                                            "type":"string",
                                            "id":
        "http://jsonschema.net/preprocessor/0/name",
                                            "required":false
                                        },
                                        "options": {
                                            "type":"object",
                                            "id":
        "http://jsonschema.net/preprocessor/0/options",
                                            "required":false,
                                            "properties":{
[0223]                                          "removeAttriIndex": {
                                                    "type":"number",
                                                    "id":
        "http://jsonschema.net/preprocessor/0/options/removeAttriIndex",
                                                    "required":false
                                                }
                                            }
                                        },
                                        "type": {
                                            "type":"string",
                                            "id":
        "http://jsonschema.net/preprocessor/0/type",
                                            "required":false
                                        }
                                    }
                                }
                            }
[0224]                  }
        }
```

[0225] 请求JSON示例


```
{
  "datasetId": "FBADDC8E-4007-4901-9CBF-328318E83DC5",
  "preprocessor": [
    {
      "name": "Remove",
      "type": "filter",
      "options": {
        "removeAttriIndex": 2
      }
    }
  ],
  "algorithm": [
    {
      "name": "Apriori",
      "options": {
        "numRules": 10
      }
    }
  ]
}
```


[0226]

[0227] 响应POST要求的字段

[0228] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0229] 响应JSON模式

[0230] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0231] 响应JSON示例

[0232] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0233] E．回归(预测)训练

[0234] URL

[0235] https://www.beulahworks.com/dm/v1/regression_train

[0236] POST请求要求的字段

[0237] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0238] 请求JSON模式

[0239] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0240] 请求JSON示例

[0241] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0242] 响应POST要求的字段

[0243] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0244] 响应JSON模式

[0245] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0246] 响应JSON示例

[0247] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0248] F．估计时间

[0249] URL

[0250] https://www.beulahworks.com/dm/v1/estimate_time

[0251] POST请求要求的字段

[0252] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0253] 请求JSON模式

```
{
        "type":"object",
        "$schema": "http://json-schema.org/draft-03/schema",
        "required":true,
        "properties":{
[0254]          "jobId": {
                    "type":"string",
                    "required":true
                }
        }
}
```

[0255] 请求JSON示例

```
{
[0256]  "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5"
}
```

[0257] 响应POST要求的字段

[0258] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0259] 响应JSON模式

```
{
  "type": "object",
[0260]
  "$schema": "http://json-schema.org/draft-03/schema",
  "id": "http://jsonschema.net",
```

```
"required": true,
"properties": {
  "estimatedFinishDate": {
    "type": "string",
    "id": "http://jsonschema.net/estimatedFinishDate",
    "required": true
  },
  "estimatedTime": {
    "type": "string",
    "id": "http://jsonschema.net/estimatedTime",
    "required": true
  },
  "jobId": {
    "type": "string",
    "id": "http://jsonschema.net/jobId",
    "required": true
  },
  "statusCode": {
    "type": "string",
    "id": "http://jsonschema.net/statusCode",
    "required": true
  },
  "statusDescription": {
    "type": "string",
    "id": "http://jsonschema.net/statusDescription",
    "required": true
  },
  "status": {
    "type": "string",
    "id": "http://jsonschema.net/status",
    "required": true,
```

[0261]

```
            "enum": [
              "success",
              "failure"
            ]
          },
          "transactionID": {
            "type": "string",
            "id": "http://jsonschema.net/transactionID",
            "required": true
          },
          "transactionTime": {
            "type": "string",
            "id": "http://jsonschema.net/transactionTime",
            "required": true
          }
        }
      }
```

[0262]

[0263] 响应JSON示例

[0264]

```
{
  "status": "success",
  "statusCode": "0",
  "statusDescription": "Success",
  "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5",
  "estimatedTime": "1 hour 30 minutes",
  "estimatedFinishDate": "2013-12-10T04:38:23:63Z",
  "transactionTime": "2013-12-10T03: 08: 23: 63Z",
  "transactionID": "241b9632-ebfb-4be2-9d6d-64910f995182"
}
```

[0265] 除与在https://www.beulahworks.com/dm/v1/classification_train.s中相同的字段之外，“jobId”是针对其估计的工作的确认，“estimatedTime”示出所选工作期间的估计时间；“estimatedFinishDate”指示所选工作将完成的估计数据和时间，如果没有发生错误的话。

[0266] G. 回叫POST

[0267] URL

[0268] https://www.beulahworks.com/dm/v1/callback

[0269] POST请求要求的字段

[0270] POST callback_url(configurable) HTTP/1.1

[0271] 内容类型:application/json

[0272] 字符集:utf-8

[0273] 接受-字符集:utf-8

[0274] 主机:callback_host (可配置)

[0275] {请求JSON}

[0276] 请求JSON模式

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "required": true,
  "properties": {
    "dataPreparationInfo": {
      "type": "object",
      "required": false,
      "properties": {
        "mode": {
          "type": "string",
```

```
            "required": true
          },
          "schemes": {
            "type": "object",
            "required": true,
            "properties": {
              "nullDataHandling": {
                "type": "string",
                "required": true
              },
              "outlierRemoval": {
                "type": "string",
                "required": true
              }
            }
          }
        }
      },
      "jobId": {
        "type": "string",
        "required": true
      },
      "modelName": {
        "type": "string",
        "required": true
      },
      "statusCode": {
        "type": "string",
        "required": true
      },
      "statusDescription": {
```

[0278]

```
        "type": "string",
        "required": true
      },
      "status": {
        "type": "string",
        "required": true,
        "enum": [
          "success",
          "failure"
        ]
      },
      "trainingInfo": {
        "type": "object",
        "required": true,
        "properties": {
          "attributeNum": {
            "type": "string",
            "required": true
          },
          "attributes": {
            "type": "array",
            "required": true,
            "items": {
              "type": "string",
              "required": false
            }
          },
          "correctlyClassifiedInstancePercentage": {
            "type": "string",
            "required": true
          },
```


[0279]

```
            "correctlyClassifiedInstancesNum": {
              "type": "number",
              "required": true
            },
            "folds": {
              "type": "number",
              "required": false
            },
            "incorrectlyClassifiedInstanceNum": {
              "type": "number",
              "required": true
            },
            "incorrectlyClassifiedInstancePercentage": {
              "type": "string",
              "required": true
            },
            "instanceNum": {
              "type": "string",
              "required": true
            },
            "scheme": {
              "type": "string",
              "required": true
            },
            "testMode": {
              "type": "string",
              "required": true
            }
          }
        }
      }
```


[0280]

[0281] }

[0282] 请求JSON示例

[0283]

```
{
  "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5",
  "status": "success",
  "statusCode": "0",
  "statusDescription": "Success.",
  "modelName": "activeCustomer_classification",
  "dataPreparationInfo": {
    "mode": "automatic",
    "schemes": {
      "outlierRemoval": "Gaussian distribution",
      "nullDataHandling": "Arithmatic mean"
    }
  },
  "trainingInfo": {
    "scheme": "weka.classifiers.rules.ZeroR",
    "instanceNum": "300",
    "attributeNum": "3",
    "attributes": [
      "764e2634",
      "852d7435",
      "279h0236"
    ],
    "testMode": "cross validation",
    "folds": 10,
    "correctlyClassifiedInstancesNum": 250,
    "correctlyClassifiedInstancePercentage": "83.3333%",
    "incorrectlyClassifiedInstanceNum": 50,
    "incorrectlyClassifiedInstancePercentage": "16.6667%"
  }
}
```


[0284]

[0285] 响应POST要求的字段

[0286] 与https://www.beulahworks.com/dm/v1/classification_train相同。

[0287] 响应JSON模式

[0288]

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "required": true,
  "properties": {
    "statusCode": {
      "type": "string",
      "required": true
    },
    "statusDescription": {
      "type": "string",
      "required": true
    },
    "status": {
      "type": "string",
      "required": true,
      "enum": [
        "success",
        "failure"
      ]
    },
    "transactionID": {
      "type": "string",
      "required": true
```

```
        },
        "transactionTime": {
          "type": "string",
          "required": true
        },
        "jobId": {
          "type": "string",
          "required": false
        }
      }
    }
```


[0289]

[0290] 响应JSON示例


```
    {
      "status": "success",
      "statusCode": "0",
      "statusDescription": "Success",
      "transactionTime": "2013-12-10T03:08:23:63Z",
      "transactionID": "241b9632-ebfb-4be2-9d6d-64910f995182",
      "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5"
    }
```


[0291]

[0292] 3. 使用API

[0293] 使用API与训练API相同,除以下之外:

[0294] 1. URL不同。“train”被替换为“use”。例如,

[0295] “https://www.beulahworks.com/dm/v1/classification_train”变为

[0296] “https://www.beulahworks.com/dm/v1/classification_use”。其它使用API也如此。

[0297] 2. “model”字段是可选的。如果未提供“model”,则系统使用大机器学习模型以执行任务。

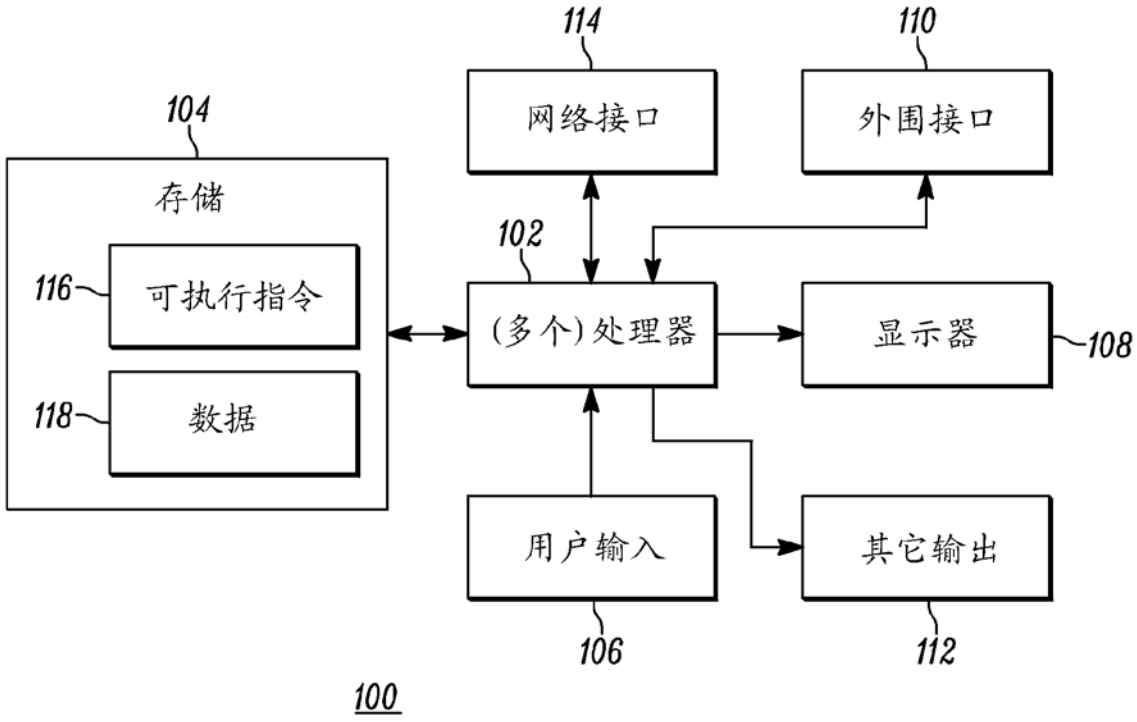

114
网络接口
110
外围接口
104
存储
102
116
可执行指令
(多个)处理器
显示器
108
118
数据
用户输入
其它输出
106
112
100

图 1

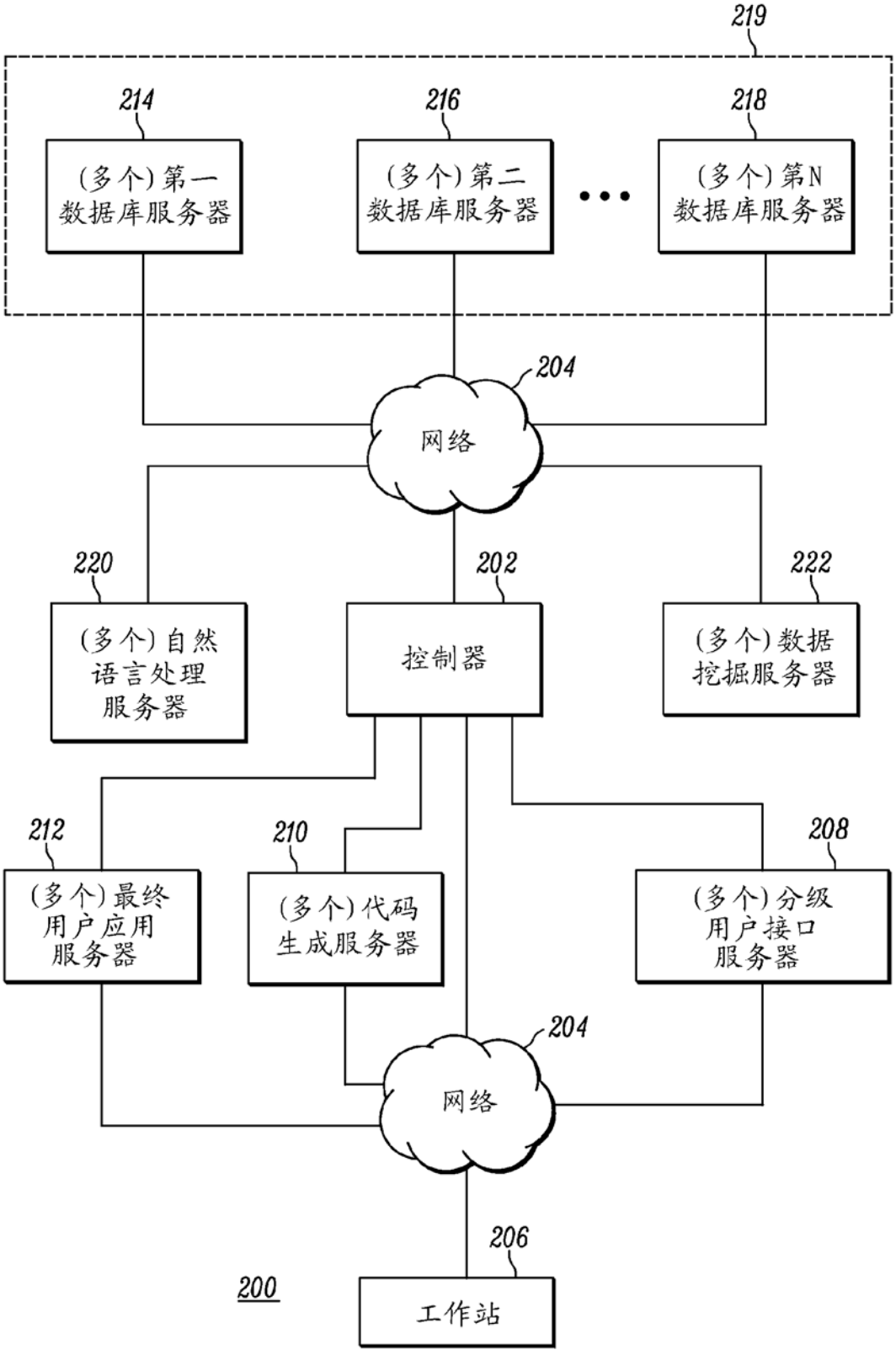

219
214
(多个)第一数据库服务器
216
(多个)第二数据库服务器
218
(多个)第N数据库服务器
204
网络
220
(多个)自然语言处理服务器
202
控制器
222
(多个)数据挖掘服务器
212
(多个)最终用户应用服务器
210
(多个)代码生成服务器
208
(多个)分级用户接口服务器
204
网络
206
200
工作站

图 2

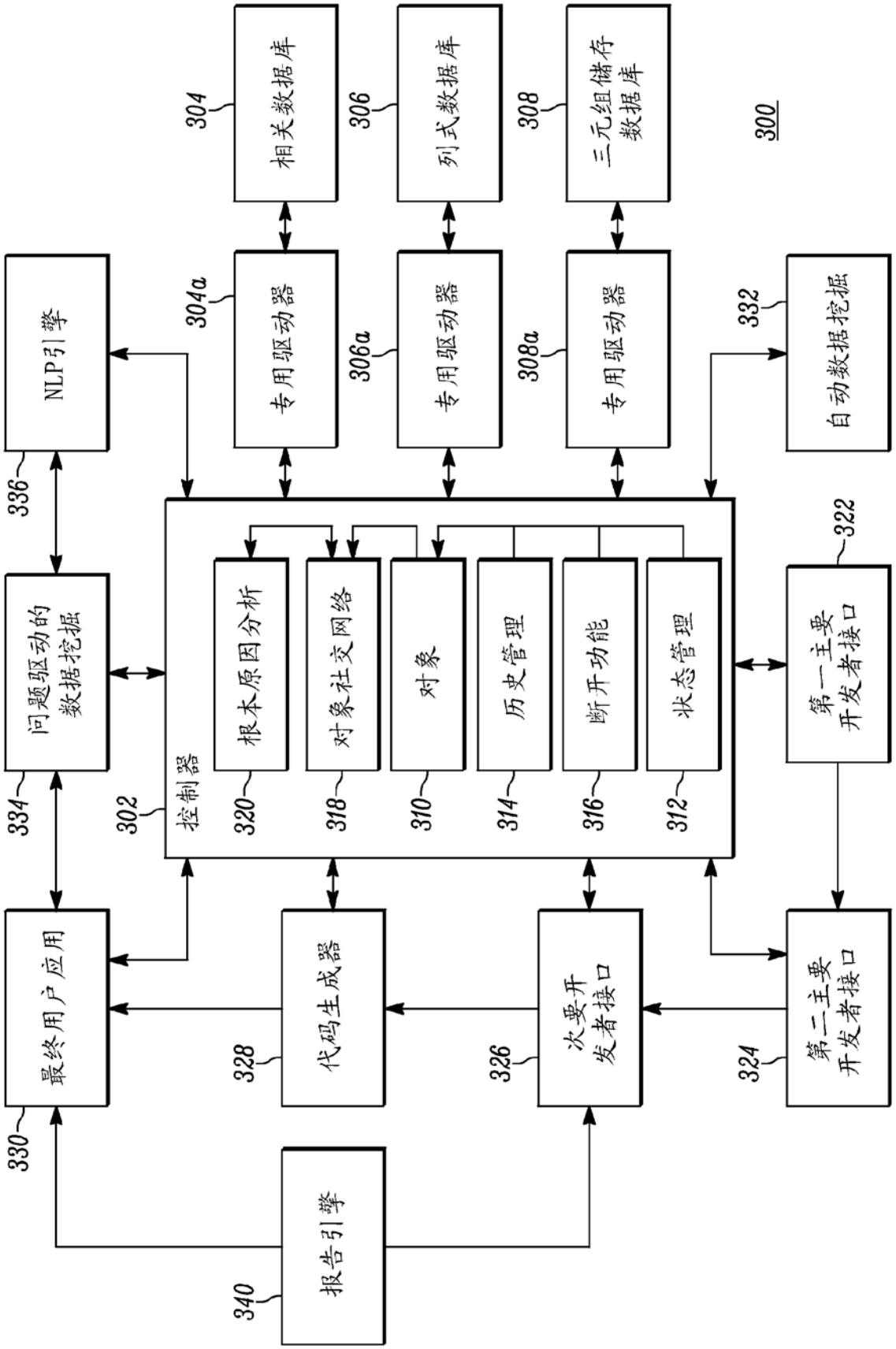

300
304
相关数据库
306
列式数据库
308
三元组储存数据库
NLP引擎
304a
专用驱动器
306a
专用驱动器
308a
专用驱动器
332
自动数据挖掘
336
问题驱动的数据挖掘
334
302
控制器
320
根本原因分析
318
对象社交网络
310
对象
314
历史管理
316
断开功能
312
状态管理
322
第一主要开发者接口
最终用户应用
330
328
代码生成器
326
次要开发者接口
324
第二主要开发者接口
报告引擎
340

图 3

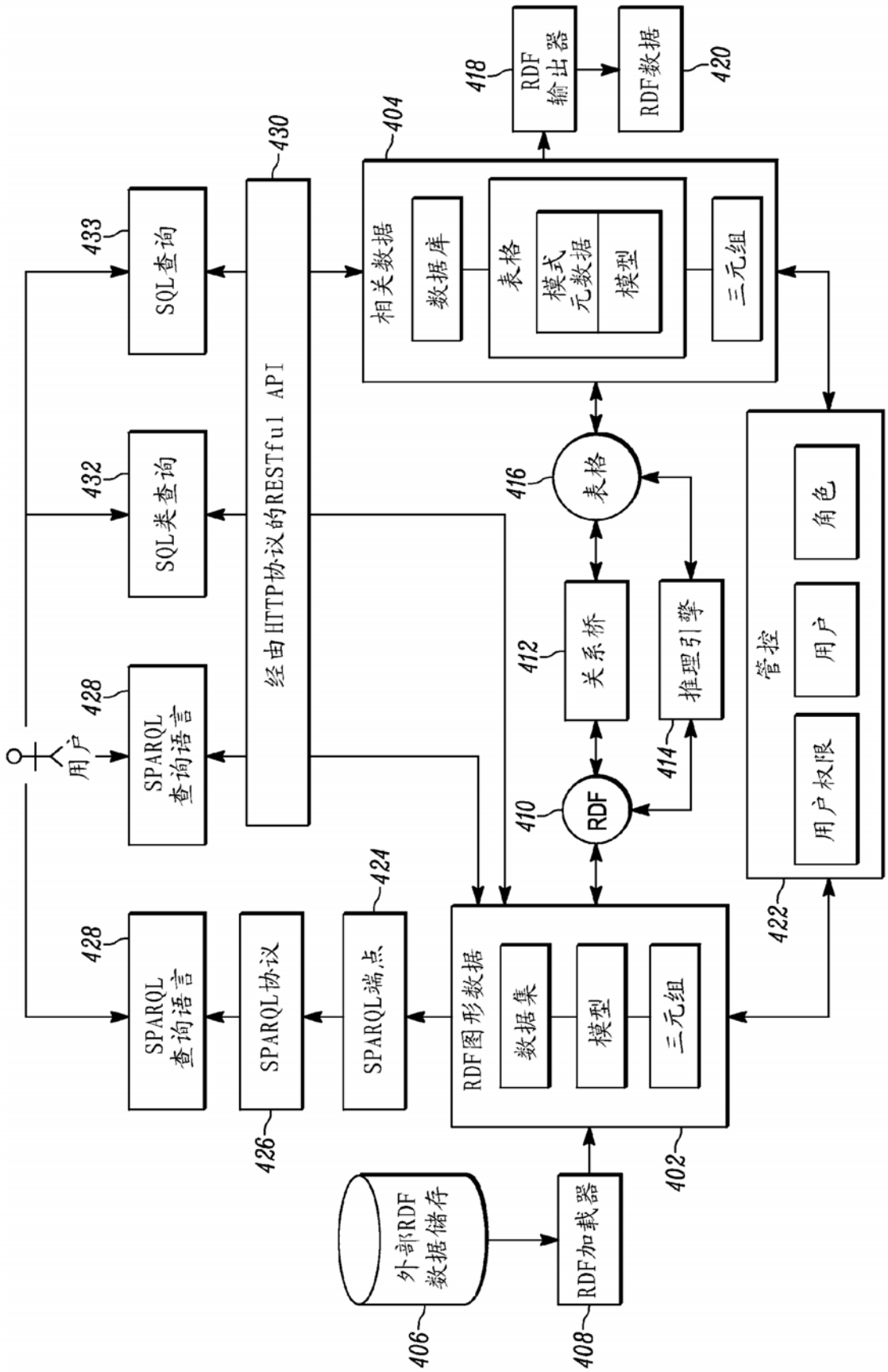

用户
SPARQL 查询语言
SQL查询
SQL类查询
SPARQL 查询语言
经由HTTP协议的RESTful API
SPARQL协议
SPARQL端点
RDF图形数据
数据集
模型
三元组
外部RDF 数据储存
RDF加载器
RDF
关系桥
推理引擎
表格
相关数据
数据库
表格
模式元数据
模型
三元组
RDF 输出器
RDF数据
管控
角色
用户
用户权限
428
433
432
428
430
424
426
406
408
410
412
414
416
404
418
420
422
402

图 4